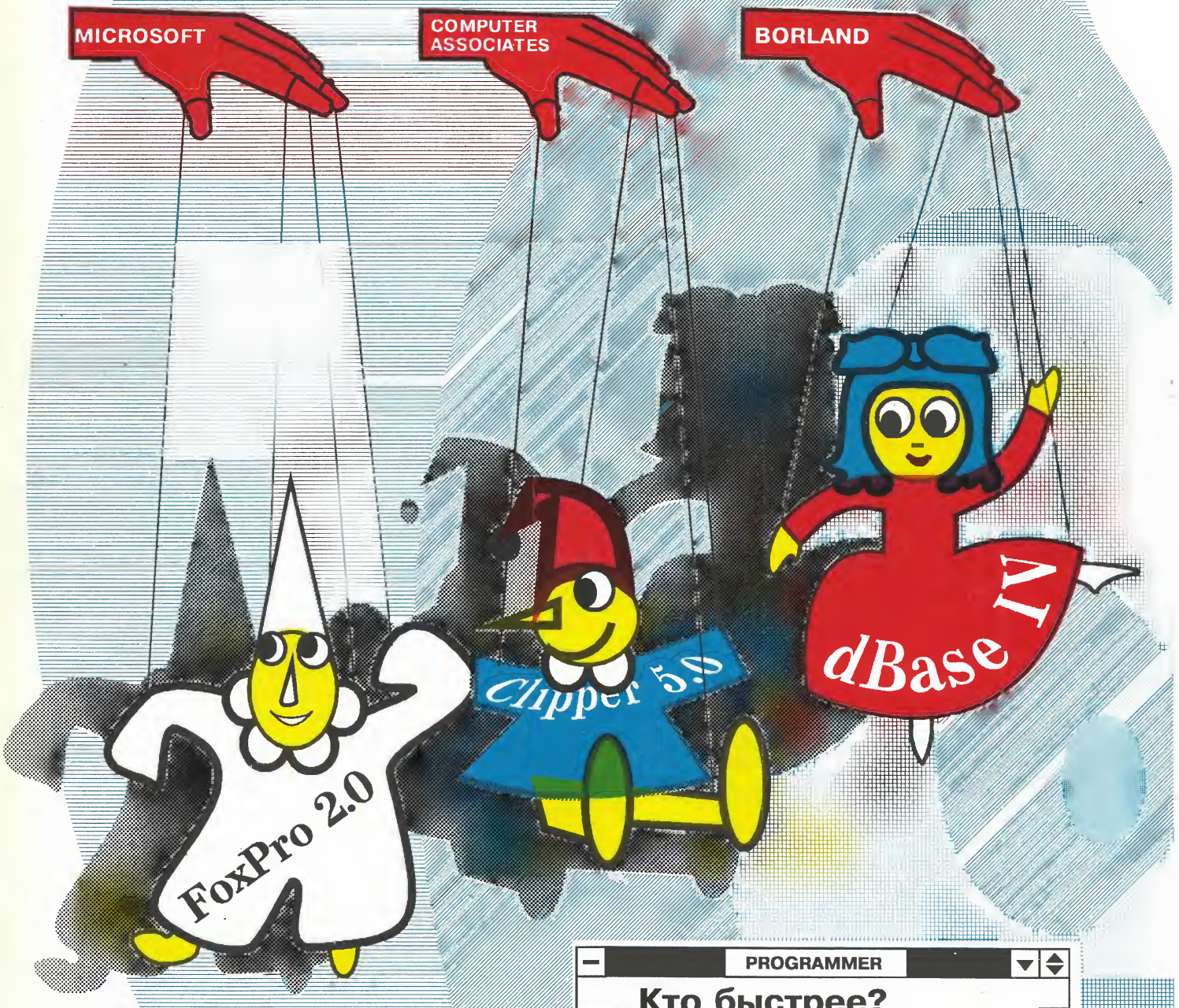
МОНИТОР

журнал для программистов-практиков

2.93

ISSN 0869-3889



PROGRAMMER

Кто быстрее?

Прямой доступ к памяти

QuickBasic-мастерам

USER

Ролевые игры

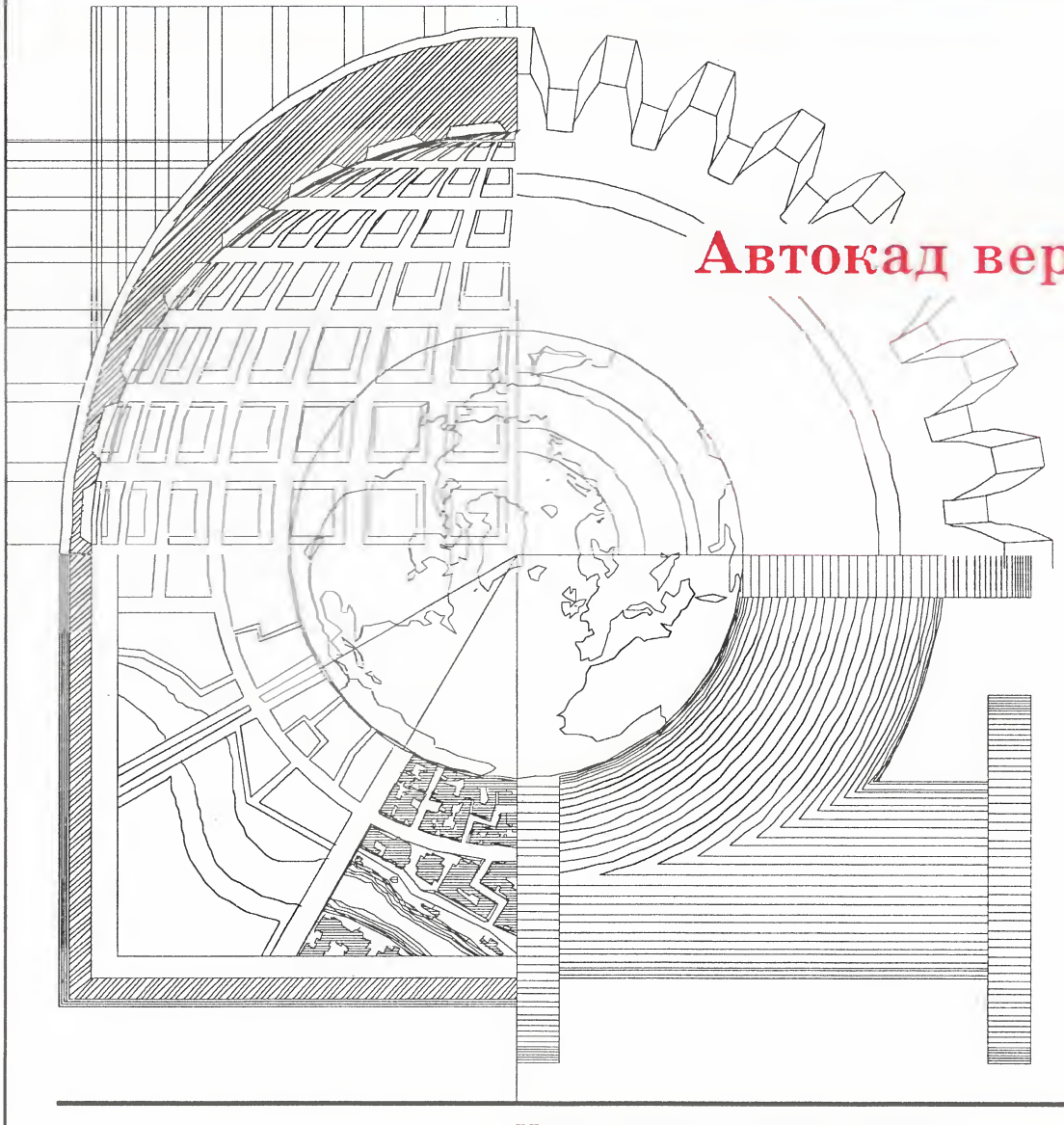
Autodesk.

Вы еще не купили Автокад?!

Обратитесь к нам и Вы получите самую свежую -
12 версию этого всемирно известного продукта.

Техническое сопровождение высочайшего уровня...
3000 приложений для всех областей проектирования...
100 учебных центров... Книги и журналы на русском языке ...

Вы никогда не будете одиноки в расширяющейся Вселенной САПР!



Автокад версии 12

Наши дилеры рядом с Вами!

Autodesk RF

Россия, 107005, Москва,
2-я Бауманская, 9 23

Тел. (095) 261 6363
Факс (095) 267 9927

Мы открываем этот номер "Монитора" рассказом о происшедших в начале 1993 года событиях, которые, по-видимому, будут определять направление развития компьютерного мира в ближайшие годы.

26 ноября 1991 года президент фирмы IBM Джон Акерс объявил о начале перестройки в IBM. Перестройка эта оказалась не идеей, а затеей. Напоминаем разницу: идея, овладевшая массами, становится материальной силой; затея, овладевшая массами, приносит неисчислимые материальные убытки. Убытки IBM достигли в 1992 году невиданной в истории американской промышленности цифры в 4,9 млрд. долларов, 40 тысяч сотрудников было уволено, акции IBM непрерывно падали. 26 января 1993 года Акерс был вынужден подать в отставку. Лидеры ведущих компьютерных фирм, которым было предложено занять освободившийся трон, в том числе Джон Скалли из Apple и Джордж Фишер из Motorola, отказались, и сейчас взойти на престол в IBM готовится табачный король из корпорации RJR Nabisco Луис Герстнер.

Причины тяжелейшего кризиса IBM кроются не только в неповоротливой бюрократической структуре. Фирма IBM создавалась и развивалась преимущественно как производитель больших машин. Переход все возрастающего числа пользователей от больших ЭВМ к сетям клиент/сервер на основе персональных компьютеров, мини-ЭВМ и рабочих станций привел к снижению в 1992 году продаж больших ЭВМ по всему миру на 16%, продажи больших ЭВМ фирмы IBM упали на 21%.

В декабре 1992 года крупнейший производитель программного обеспечения для локальных сетей фирма Novell объявила о приобретении Unix System Laboratories — разработчика операционной системы Unix. Три месяца спустя основные производители операционных систем — диалектов Unix, а именно Sun Microsystems, Novell, Santa Cruz Operation и примкнувшие к ним IBM и Hewlett-Packard составили альянс, цель которого — выработка общего подхода к интерфейсу и основным элементам Unix. В результате должна существенно упроститься разработка прикладных программ в среде Unix, а сами программы будут выглядеть для пользователя одинаково, вне зависимости от диалекта Unix, с которым работает его компьютер.

В мае ожидается выпуск операционной системы Windows NT фирмы Microsoft, а в 1994 году — объектно-ориентированной операционной системы Taligent, совместного проекта IBM и Apple. Все это предвещает в ближайшие несколько лет ожесточенную борьбу за передел рынка операционных систем между Windows NT, OS/2, Unix и, возможно, Taligent и NextStep. Шансы последней уменьшились после того как союзники из возглавляемого Novell альянса отказались от сотрудничества с фирмой Next. Не сдается и DOS, очередную версию которой Microsoft передал для испытаний фирмам-разработчикам программного обеспечения.

В течение последних двух лет цены на персональные компьютеры в США снижались ежегодно на 40%. Падение цен продолжается: 8 марта один из крупнейших производителей персональных компьютеров Compaq объявил о снижении цен на свои компьютеры и принтеры в США на 20%. Теперь американский покупатель может купить персональный компьютер с процессором i486, 4 Мбайтами памяти и жестким диском емкостью 120 Мбайтов менее чем за 1000 долларов.

22 марта 1993 года Intel Corporation объявила о начале промышленных поставок микропроцессора Pentium. RISC процессор с тактовой частотой 66 МГц и производительностью 112 миллионов команд в секунду содержит более 3 миллионов транзисторов и предоставляет пользователям качественно новый уровень вычислительной мощности. Поставки Pentium в 1993 году будут невелики — всего несколько десятков тысяч процессоров. Однако можно с уверенностью предполагать, что через 1-2 года в мире будут продаваться миллионы персональных компьютеров на базе Pentium по цене, не превосходящей цены сегодняшних систем на базе i486.

Нам же предстоит запастись терпением и немного подождать светлого будущего, когда, переведя полученные в кассе рубли по курсу Центрального Банка, мы обнаружим, что искомые 1000 долларов составляют одну треть от месячной зарплаты. Зато уж тогда мы будем вознаграждены восьмым (десятым, двенадцатым) поколением микропроцессоров Intel с производительностью миллиарды (десятки миллиардов, триллионы) операций в секунду. Если, конечно, не помешает световой барьер...

Журнал для программистов-практиков

Редколлегия:
Д.Э.Бройтман (главный редактор)
В.В.Водолазский
В.С.Головин
Н.Н.Косарев
А.М.Миляков (ответственный секретарь)
Н.А.Никитина (отдел рекламы)
А.В.Орехов (издательский директор)
Е.В.Равич
Л.И.Турок

Графическое бюро:
С.В.Макельский (главный художник)
Г.Л.Карчикян (компьютерная верстка)

Отдел распространения:
Р.А.Болдырев
А.А.Винокуров
Е.В.Гун

Представительство в Европе (Германия)
Марина Гетманская
тел. 8-(0631)-7-49-05

В оформлении номера использованы рисунки
А.С.Замулы

Подписано в печать с оригинал-макета 05.04.93
Формат 60x84/8.

Печать офсетная. Печ. л. 10,0
Цена договорная
Отпечатано в ИПК "Московская правда", Москва, ул. 1905 года, д. 7. Тираж 30000. Заказ 2588

Издатель:
Научно-техническое общество
Софт·Москва

Для писем: 117419 Москва, а/я 765
Тел. (095) 238-99-33
Факс (095) 238-32-78
Телекс 613322 GRANT SU

© "Монитор", 1993

Технологии программирования

М. Трофимов. **Кто быстрее?** 4

Статья посвящена вопросам оптимизации скорости исполнения программ.

А. Колесов
О. Павлова
Профессиональная работа в среде Microsoft QuickBASIC Часть 1. Технология разработки программ 12

Первая часть цикла статей о Quick Basic. Описаны структура Basic программы, форматы программных модулей, общая схема разработки программ, использование библиотек подпрограмм.

Базы данных

М. Корнеев. **СУБД FoxPro 2.0 — новый лидер семейства xBase** 18

Рассматриваются новые возможности, появившиеся в версии 2.0 СУБД FoxPro.

Ю. Сюганов. **Компиляция программ на языке dBase IV** 24

Теперь Вы можете создавать базы данных и программы в интерпретирующей среде dBase IV, а затем компилировать их, получая скорость и гибкость откомпилированной системы.

А. Карев. **Расширенная сетевая модель данных MDBS III: плюсы и минусы** 30

Статья рассказывает про сетевую модель данных и ее реализацию в СУБД MDBS III.

А. Ермолов. **Программа для работы с библиотеками процедур PAL** 36

Описание диалоговой программы поддержки библиотеки процедур СУБД PARADOX. Приведен исходный текст библиотеки на языке PAL.

Windows

Д.Пластинин. **Многооконный интерфейс в среде MS Windows** 38

Обсуждение проблем, связанных с разработкой MDI в среде Windows. Примеры реализации на языке C++ 3.1.

DOS под микроскопом

О. Шарапов. **Программная русификация матричных принтеров** 48

Варианты русификации принтеров с различной внутренней организацией. Приведен пример русификатора для принтера Star LC-10, реализованный на языке ассемблера.

DOS под микроскопом

А. Аброськин. **Игры и как с ними бороться** 56

Описывается резидентная программа-перехватчик игр.

В. Аношин. **Что же там, в загрузочном секторе?** 60

Детальное описание процедуры загрузки операционной системы. Приведен код программы-загрузчика на языке ассемблера.

Бизнес

Компания "Статус" - первое знакомство 63

Компьютер в разрезе

В.Плотников. **Программирование контроллера прямого доступа к памяти** 64

Описана методика использования контроллера ПДП i8237a при организации обмена данными с периферийными устройствами. Приведены примеры процедур на языке ассемблера.

Ю. Косивцов. **Заметки дилетанта** 69

Описывается небольшая программа на ассемблере - "противоядие" от изменения области данных BIOS, относящейся к видеоадаптеру.

Письма

Ответ сердитому читателю 71

Игры

Игры с несколькими участниками Часть 1 74

Анализируется популярный жанр компьютерных игр.

Книги

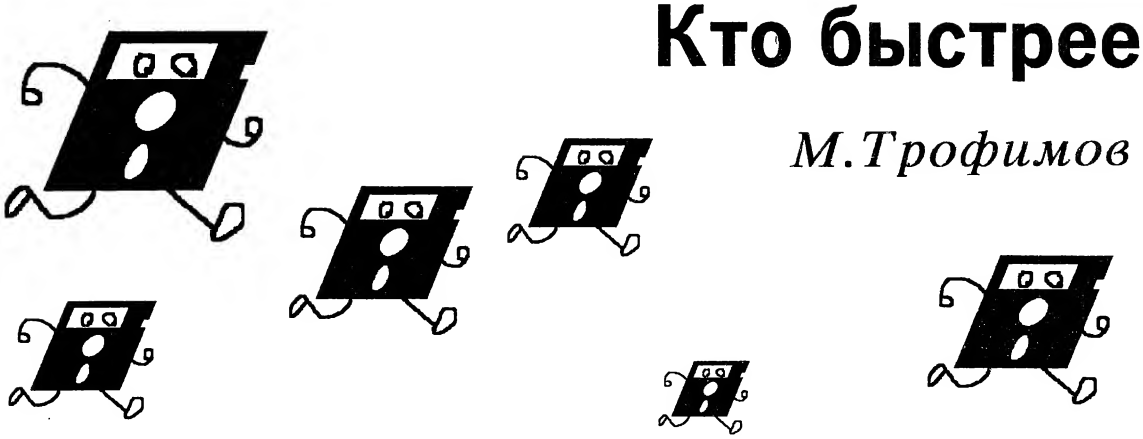
В. Водолазкий. **Рецензия на книгу В.Ю. Романова "Популярные форматы файлов для хранения графических изображений на IBM PC"** 76

Аппаратура

Caviar приходит в Россию 78

Кто быстрее?

М.Трофимов



Всевозможные испытания на скорость очень популярны. Нередко их результаты используют в рекламных целях. Однако, к сожалению, далеко не всегда такая реклама соответствует истине. Вы можете убедиться на собственном опыте, сколь непросто получить по-настоящему убедительные результаты.

Кто быстрее? Вернее, "что быстрее?" — такой вопрос в программировании возникает очень часто. Иногда теоретическую трудоемкость того или иного алгоритма возможно строго оценить на бумаге. Но и в этом случае программисту-практику бывает недостаточно знать, что данный алгоритм является, к примеру, "экспоненциальным" по времени выполнения. Не менее, а зачастую более важны конкретные значения экспоненциальных множителей в имеющейся вычислительной среде. Строгий анализ алгоритма, доказательство его правильности и оценка трудоемкости возможны далеко не всегда, и очень часто математически строгие оценки подменяются общими рассуждениями, что "что-то должно работать быстрее, а что-то медленнее". Эти рассуждения уже породили не один программистский миф [1]. Поэтому экспериментальное сравнение скоростей выполнения двух или нескольких программ и программных фрагментов бывает далеко не лишним и часто приводит к неожиданным результатам.

В принципе нет ничего сложного в том, чтобы измерить время выполнения какого-либо фрагмента. Сравним, например, время, затраченное на исполнение следующих операторов:

```
1)   i := i + 1;
2)   inc ( i );
```

Для этого напишем испытательную программу, например, такую, как приведенная в листинге 1. Естественно, что в результате ее работы мы получим полную чепуху, поскольку измеряемые операторы исполняются слишком быстро. Поэтому для каждого измерения фрагмент надо исполнить многократно:

```
for   j:= i to 1600000 do
      i:= i+1;
```

Сделав необходимые изменения, получаем программу, результаты работы которой приведут к утверждению, что первый фрагмент быстрее второго. Это довольно неожиданный результат, так как в инструкции на Турбо Паскаль утверждается обратное. Но не будем торопиться с выводами, а оптимизируем программу, отключив тест Run-Time системы Турбо Паскаля (опция $R-). Вот теперь полученный результат будет соответствовать утверждению инструкции.

Даже из этого тривиального примера видно, что далеко не всегда возможно в общем случае ответить на вопрос "что быстрее?". Действительно, если программист собирается достаточно долго эксплуатировать свою программу, не отключая Run-Time тестов, поскольку только для очень простых программ можно гарантировать стопроцентную корректность со стопроцентной уверенностью, то обоснованное предпочтение может быть отдано первой реализации, если же планируется оптимизация кода, то второй. В большинстве случаев выигрыш во времени может оказаться столь незначительным, что произвольная реализация будет наиболее оптимальной, поскольку сэкономит время и силы самого программиста, предотвратив их растрату на несущественные детали. Поэтому практический смысл имеет не тотальная оптимизация всей программы, а лишь наиболее критичных по времени участков.

Многие программы подчиняются правилу: на исполнение 90% кода тратится лишь 10% процессорного времени (остальные 90% времени тратятся на исполнение всего лишь 10%). Для развитых программ, снабженных удобным пользовательским интерфейсом, может оказаться, что на исполнение 99% их кода тратится лишь 1% общего времени! Критические участки можно определить с помощью профиля времени выполения программы, полученного экспериментально. Для таких измерений существует много удобных инструментов, например, Turbo Profiler фирмы Борланд. Чаще же всего программист и так знает наиболее критичные места своей программы: это, как правило, процедуры, применяющие трудоемкие алгоритмы, действия, связанные с перебором, возвратом, последовательное приближение и так далее.

Второй важный момент, который можно обнаружить даже на нашем простеньком примере, это несовпадение результатов измерений для разных ПК с разной тактовой частотой процессора. Вычислительное окружение, например, наличие в памяти резидентных программ, тоже может повлиять на ход измерений.

Архитектурные особенности ПК также могут влиять на результаты измерений и в некоторых случаях даже на их воспроизводимость! Интересный пример такой ситуации описан в [2].

До сих пор мы говорили о простейшем случае испытаний на скорость: соревнование двух программных фрагментов, написанных для одного и того же компилятора, на одном и том же компьютере, в одном и том же системном окружении. Это лишь одна узкая область из гигантского спектра различных компьютерных гонок: сравниваются скорости компиляции для различных компиляторов одного и того же языка программирования и скорости работы одинаковых программ, написанных на разных языках, скорости работы прикладных программ под управлением разных операционных систем на одном компьютере и скорости работы разных компьютеров. По пословице "чем дальше в лес — тем больше дров": количество аргументов у функции скорости (если, конечно, в каком-либо из этих случаев можно говорить о функции!) возрастает. Как можно сравнивать несравнимое: устроить соревнования автомобиля "Формула-1", предназначенного для специального гоночного трека, и гусеничного трактора повышенной проходимости для таежного бездорожья? Но в компьютерных областях подобные гонки оказывают значительное влияние на их прогресс, несмотря на свою крайнюю нестрогость, а поэтому стали традиционными как для науки, так и для инженерии. Особо следует отметить распространенность всевозможных испытаний на скорость в сферах компьютерного бизнеса. И тут, к сожалению, очень часто желаемое выдается за действительное. И это при том, что существует огромное количество стандартных испытательных программ, типа знаменитого Whetstone теста, и распространенных подходов, типа измерения в MIPS (миллионах инструкций в секунду). Например, [3], оператор

```
A := B + C
```

для компьютера VAX 11/780 может быть представлен одной инструкцией вида:

```
сложить адрес_B, адрес_C, адрес_A
```

В то же время для RISC процессора требуются четыре инструкции:

```
загрузить адрес_B, регистр_1
загрузить адрес_C, регистр_2
сложить   регистр_1, регистр_2, регистр_3
запомнитьрегистр_3, адрес_A
```

Конечно, данный пример слишком избит и примитивен, однако он хорошо иллюстрирует суть проблемы, детальное обсуждение которой следует искать в специальной литературе: разработано множество подходов, и при том, что для каждого вида "гонок" они несколько различаются, такие упомянутые нами принципы, как воспроизводимость результатов и учет параметров, влияющих на измерения, остаются общими для всех них. Поэтому, когда мы хотим сделать какое-либо сравнение, нужно готовиться не к одному-единственному измерению, а к серии однотипных испытаний — измерительных экспериментов, причем условия проведения и результат каждого такого эксперимента должны быть четко и безошибочно задокументированы. Гарантировать последнее могут специальные программные средства поддержки эксперимента (использование таких средств, пожалуй, главная технологическая хитрость для успешной работы). В зависимости от задачи, целей и методики эти средства могут быть разными и во многих случаях пишутся программистом самостоятельно, применительно к конкретной работе. Не стоит жалеть времени на их написание и отладку: потраченное время компенсируется тем, что исчезнет необходимость утомительной переписи цифр вручную с экрана компьютера (а потом ввода их с клавиатуры все в тот же компьютер для последующей обработки!), кроме этого, у Вас будет уверенность, что Вы не ошиблись при переписывании, а это тоже чего-нибудь стоит!

В качестве примера такой полезной самоделки, иллюстрирующей know-how, далее приводится шаблон-программа BMT (BenchMarkTest). Она предназначена для поддержки экспериментов по сравнению скоростей работы программных фрагментов типа тех, что мы уже обсуждали в начале статьи.

BMT написана на Турбо Паскале 5.0, но ее можно приспособить и для более старших версий: 5.5 и 6.0. В этой несложной программе применен ряд приемов аппаратурного и системнозависимого программирования на Турбо Паскале, а также примыкающие к ним приемы условной компиляции.

Исходный текст программы BMT разбит на несколько файлов: BMT.PAS (листинг 2), BMTEST.CNT (листинг 3), BMTEST.VAR (листинг 4) и BMTEST.PRC (листинг 5). Все предназначенные для изменений строки находятся в файле-шаблоне BMT.PAS. Остальные неизменяемые файлы подключаются к нему с помощью опции "вставка файла"

```
{$I имя файла}
```

компилятора Турбо Паскаля. Файл BMTEST.CNT содержит константы, BMTEST.VAR — переменные, а BMTEST.PRC — процедуры. Все определяемые в этих файлах глобальные имена начинаются со специального префикса SYSBMT_000_, чтобы избежать случайного совпадения с именами, определяемыми пользователем. В шаблоне отведено место для определения констант пользователя, его начало помечено комментарием

```
{User const}
```

(поскольку при испытаниях драйвер кириллицы может оказаться нежелательным, все комментарии в тексте программы лучше писать по-английски), за этой строкой пользователь может расположить свои константы. Аналогично отводится место под типы, переменные и процедуры пользователя, а в теле прог-

раммы отведено место под операторы сравниваемых фрагментов. Каждый из фрагментов носит имя Test с соответствующим номером. Программы, сделанные из BMT, поддерживают и документируют серию измерений, состоящую максимум из десяти экспериментов (исполнений BMT программы). Каждое такое исполнение приводит к созданию файлов с именем

BMTEST-номер_эксперимента.расширение
Здесь номер_эксперимента —0...9;
расширение — LOG, DOC и DAT.

В файл с расширением LOG заносится вся информация о параметрах вычислительной среды, а в файл DOC заносятся результаты измерений. Предусмотрено (листинг 3) два формата организации DOC файла: вертикальный (vertical) и горизонтальный (horizontal). В первом случае данные располагаются построчно:

```
<номер теста (8 позиций)> <время исполнения (8 позиций)>
```

Во втором — в одну строку:

```
<время исполнения 1 теста> <время исполнения 2 теста> <...>
```

И тот и другой формат удобен для обработки с помощью программ электронных таблиц типа Lotus 1-2-3, Quattro и так далее.

В файл с расширением DAT заносятся результаты вычислений, произведенных тестами. Эти результаты нужны для контроля правильности работы программы. Контроль осуществляется путем сравнения DAT файла, полученного в данном эксперименте, с аналогичным файлом BMTESTDT.DAT, полученным в результате первого эксперимента (таким образом, в первом эксперименте проверка не производится, имя DAT файла — BMTESTDT.DAT, а не BMTEST-1.DAT).

В командной строке BMT программы предусмотрено 2 параметра: первый состоит из опций '?', '+', '0' в любой последовательности, второй — имя файла листинга. Смысл опций следующий:

+ — распечатать листинг в файл листинга;
? — вывести help информацию (содержимое BMTEST.HLP файла — текстовый файл — пишется пользователем) на экран и прекратить работу;
0 — установить номер эксперимента равным нулю (вне зависимости от номера предыдущего эксперимента).

Если опция "0" не установлена, то нумерация экспериментальной серии начнется с номера 1.

Распечатка листинга — это копирование BMT программы в файл листинга (по умолчанию BMTEST.LST). Предусмотрены следующие команды управления листингом:

{.-} — выключение листинга;
{.+} — включение листинга.

Эти команды располагаются непосредственно в тексте BMT программы. Для установки контроля за окружением предусмотрена фиксация конфигурации DOS в файлах CONFIG.BMT и AUTOEXEC.BMT. Файл AUTOEXEC.BMT включает строку вида:

set BNCHMARK = произвольная последовательность символов, задаваемая пользователем.

(Рекомендуется указать фирму — изготовитель ПК, тактовую частоту и производительность ЦП.)

При исполнении EXE файла BMT программа ищет строку BNCHMARK в окружении DOS. Если эта строка не определена, то BMT (с согласия пользователя) определяет драйвер диска, с которого был загружен COMMAND.COM (системный диск), и переименовывает находящиеся в корневой директории файлы AUTOEXEC.BAT и CONFIG.SYS в файлы AUTOEXEC.B-M и CONFIG.B-M. Если такое переименование состоялось, то в корневую директорию копируются файлы AUTOEXEC.BMT и CONFIG.BMT под именами AUTOEXEC.BAT и CONFIG.SYS соответственно.

Потом автоматически производится перезагрузка DOS. Так как в AUTOEXEC.BAT теперь присутствует имя BMT программы (добавляется при копировании AUTOEXEC.BMT), то эта программа автоматически запускается после перезагрузки.

После выполнения BMT программы с согласия пользователя производится поиск файлов AUTOEXEC.B-M и CONFIG.B-M в корневой директории системного диска. В случае если эти файлы найдены, из этой директории удаляются файлы AUTOEXEC.BAT и CONFIG.SYS, файлам AUTOEXEC.B-M и CONFIG.B-M возвращаются их первоначальные имена и перезагружается DOS.

При создании новой BMT программы (добавлением в шаблон BMT.PAS интересующих вас фрагментов и определений для использованных в них имен) нужно выбрать подходящее значение константы SYSBMT_0000_TestLoopFinal, определяющее количество повторных исполнений для каждого теста. Значение считается подходящим, если время исполнения тестов лежит в интервале, задаваемом константами SYSBMT_0000_MinTime и SYSBMT_0000_MaxTime. В противном случае на экран выводятся сообщение об ошибке и рекомендуемое значение SYSBMT_0000_TestLoopFinal, причем, если время выполнения какого-либо теста оказалось меньше, чем SYSBMT_0000_MinTime, то выполнение EXE файла BMT программы прекращается. Вы можете изменить также и названные константы, определяющие интервал допустимого времени, при этом следует учитывать, что при слишком быстром исполнении тестов погрешность измерений может возрасти. ■

Литература

1. М.И.Трофимов. Итерация против рекурсии.//Монитор, № 8, 1992.
2. Р.Г.Герр. Кто виноват?//Мир ПК, № 1, 1991.
3. R.P.Weicker. An Overview of Common Benchmarks.//IEEE Computer, V. 23, № 12, 1990.

Листинг 1. Испытательная программа для сравнения скорости выполнения i := i + 1 и inc (i)

```
program test1;
var
    Timer : longint absolute $0040:$006C;
    i : integer;
    TimeStart, TimeStop : longint;
begin
    i := 0;
    TimeStart := Timer;
    i := i + 1;
    TimeStop  := Timer;
    writeln ( '1) ', TimeStop - TimeStart );
    i := 0;
    TimeStart := Timer;
    inc ( i );
    TimeStop := Timer;
    writeln ( '2) ', TimeStop - TimeStart );
end.
```

Листинг 2. BMT.PAS файл: шаблон–программа

```
{*****************************************************}
{                                                     }
{       Benchmark pattern.                            }
{       Written by Michael Trofimov, 1991.            }
{       Turbo Pascal 5.0                              }
{                                                     }
{*****************************************************}
{.-}
{ 1. Compiler options : }
{ > > } {$A+}     { -allign data              }   { <-for change }
{ > > } {$B+}     { -Boolean evalution        }   { <-for change }
{ > > } {$E-}     { -emulation 8087           }   { <-for change }
{ > > } {$I+}     { -I/O checking             }   { <-for change }
{ > > } {$N+}     { -numeric processing       }   { <-for change }
{ > > } {$R-}     { -range-checking           }   { <-for change }
{ > > } {$S-}     { -stack-overflow checking  }   { <-for change }
{ > > } {$V-}     { -var-string checking      }   { <-for change }
    {$IFOPT I+} {$DEFINE SYSBMT_0000_IOCHK} {$ENDIF   }

{ 2. The program }
program BenchMark;
Uses Dos, Crt;
Const
    {$I BMTEST.CNT }
{ > > } SYSBMT_0000_CompilerNm = 'TURBO';
{ > > } SYSBMT_0000_DefaultParm = '';              { <-for change }
{ > > } SYSBMT_0000_Title = 'Benchmark test: ...'; { <-for change }

    { Time of test: }
{ > > } SYSBMT_0000_TestLoopFinal = 320;           { <-for change }
{ > > } SYSBMT_0000_MinTime = 1820; { = 100 sec }  { <-for change }
{ > > } SYSBMT_0000_MaxTime = 36400; {≈ 30 min }  { <-for change }
{ > > } SYSBMT_0000_AvrTime = 3000; { average }   { <-for change }
{.+}
    { User const.: }
{ > > }                                           { <-for insert }

type
    SYSBMT_0000_dummy = Boolean;
    { User type : }
{ > > }                                           { <-for insert }
var
                                  {.-}
                                  {$I BMTEST.VAR }
{.+}
    { User var.: }
{ > > }                                           { <-for insert }
                                  {.-}
                                  {$I BMTEST.PRC }
{.+}

{ > > } { User procedures : }                     { <- for insert }

begin { program body }
                          {.-} SYSBMT_0000_Start;
{.+}
{=====================================}

{ TEST 1: }
                          {.-} SYSBMT_0000_WaitMess;
                               SYSBMT_0000_TimeStart :=
                               SYSBMT_0000_Timer;
                          for  SYSBMT_0000_TestLoopControl := 1 to
                               SYSBMT_0000_TestLoopFinal do
{.+}
{ > > }    {---- test 1 : }
{---- end of test 1.}
                          {.-} SYSBMT_0000_TimeStop :=
                               SYSBMT_0000_Timer;
                               SYSBMT_0000_TimeOut ( 1,
                               SYSBMT_0000_DocFile );
{.+}
{ > > } writeln ( SYSBMT_0000_DatFile, '...' );   { <-for change }

{ TEST 2: }
                          {.-} SYSBMT_0000_WaitMess;
                               SYSBMT_0000_TimeStart :=
                               SYSBMT_0000_Timer;
                          for  SYSBMT_0000_TestLoopControl := 1 to
                               SYSBMT_0000_TestLoopFinal do
{.+}
{ > > } {---- test 2 : }
{---- end of test 2.}
                          {.-} SYSBMT_0000_TimeStop :=
                               SYSBMT_0000_Timer;
                               SYSBMT_0000_TimeOut ( 2,
                               SYSBMT_0000_DocFile );
{.+}
{ > > } writeln ( SYSBMT_0000_DatFile, '...' );   { <-for change }

{=====================================}
{.-}                      SYSBMT_0000_Finish;
end.
```

Листинг 3. BMTEST.CNT файл: константы

```
{*****************************************************}
{                                                     }
{       Benchmark pattern : insert file BMTEST.CNT    }
{       See main file BMT.PAS.                        }
{       Written by Michael Trofimov, 1991.            }
{       Turbo Pascal 5.0                              }
{                                                     }
{*****************************************************}

{ Options }
SYSBMT_0000_ListO = '+';
SYSBMT_0000_ZeroO = '0';
SYSBMT_0000_HelpO = '?';
SYSBMT_0000_EnvNm = 'BNCHMARK';

{ Data format: }
SYSBMT_0000_horizontal    = true;
SYSBMT_0000_vertical      = false;

{ File names: }
SYSBMT_0000_DatName = 'BMTESTDT';   SYSBMT_0000_DatExt = '.DAT';
SYSBMT_0000_DocName = 'BMTEST-';    SYSBMT_0000_DocExt = '.DOC';
SYSBMT_0000_LogName = 'BMTEST-';    SYSBMT_0000_LogExt = '.LOG';
                                    SYSBMT_0000_TempExt = '.TMP';
SYSBMT_0000_HelpName        = 'BMTEST.HLP';
SYSBMT_0000_DefListingNm    = 'BMTEST.LST';
```

Листинг 4. BMTEST.VAR файл: переменные

```
{************************************************}
{                                                }
{       Benchmark pattern : insert file BMTEST.VAR }
{       See main file BMT.PAS.                   }
{       Written by Michael Trofimov, 1991.       }
{       Turbo Pascal 5.0                         }
{                                                }
{************************************************}

{ Timer variables: }
SYSBMT_0000_Timer : longint absolute $0040:$006C;
SYSBMT_0000_TimeStart,
SYSBMT_0000_TimeStop,
SYSBMT_0000_TestLoopControl : longint;

{ System clock: }
SYSBMT_0000_Hour, SYSBMT_0000_Min, SYSBMT_0000_Sec,
SYSBMT_0000_Sec100,                    { for GetTime proc.}
SYSBMT_0000_Year, SYSBMT_0000_Month, SYSBMT_0000_Day,
SYSBMT_0000_DayOfWeek : word;          { for GetDate proc.}

{ Files: }
SYSBMT_0000_ProgPath : PathStr;
SYSBMT_0000_ProgDir : DirStr;
SYSBMT_0000_ProgExt : ExtStr;
SYSBMT_0000_DocNm,
SYSBMT_0000_LogNm,
SYSBMT_0000_ProgName : NameStr;
SYSBMT_0000_ListingNm : String [12];
SYSBMT_0000_SampleFile, SYSBMT_0000_TempFile,
SYSBMT_0000_DatFile, SYSBMT_0000_DocFile,
SYSBMT_0000_LogFile : text;

{ other service variables: }
SYSBMT_0000_experiment : char;
SYSBMT_0000_CompFlag, SYSBMT_0000_ListFlag,
SYSBMT_0000_NZeroFlag, SYSBMT_0000_HelpFlag : Boolean;
SYSBMT_0000_ver : word;                { DOS version }
SYSBMT_0000_EnStr : string;
SYSBMT_0000_DataFormat : Boolean;

{++++++++++++++++++++ eof BMTEST.VAR ++++}
```

Листинг 5. BMTEST.PRC файл: процедуры

```
{************************************************}
{                                                }
{       Benchmark pattern : insert file BMTEST.PRC }
{       See main file BMT.PAS.                   }
{       Written by Michael Trofimov, 1991.       }
{       Turbo Pascal 5.0                         }
{                                                }
{************************************************}

procedure SYSBMT_0000_beep;
var
    i : byte;
begin
    for i := 1 to 6 do
        begin
            Sound ( 200 * i );
            Delay (40);
            Sound ( 2200 div i );
            Delay (30);
        end;
    NoSound;
end; {_____ beep _____}

procedure SYSBMT_0000_WaitMess;
begin
    TextColor ( White + Blink );
    write ( 'Wait, please ! ');
    TextColor ( LightGray );
end; {_____ SYSBMT_0000_WaitMess _____}

procedure SYSBMT_0000_error ( s : string );
begin
    writeln ( s );
    SYSBMT_0000_beep;
    halt ( 1 );
end; {_____ SYSBMT_0000_error _____}

function SYSBMT_0000_exist (name : DirStr) : Boolean;
                                    { Does this file exist ? }
var
    local : text;
    flag : Boolean;
begin
    assign ( local, name );
    {$I-}
    reset (local);
    {$IFDEF SYSBMT_0000_IOCHK} {$I+} {$ENDIF}
    flag := IOResult = 0;
    if flag then
        close ( local );
    SYSBMT_0000_exist := flag;
end; {_____ SYSBMT_0000_exist _____}

procedure SYSBMT_0000_OpenFiles;
begin
{ Define number of current experiment: find previous *.DOC & *.LOG files }
    SYSBMT_0000_experiment := '0';

    if SYSBMT_0000_NZeroFlag then
    repeat
        SYSBMT_0000_experiment := succ ( SYSBMT_0000_experiment );
    until not ( SYSBMT_0000_exist ( SYSBMT_0000_DocName +
            SYSBMT_0000_experiment + SYSBMT_0000_DocExt )
        and
            SYSBMT_0000_exist ( SYSBMT_0000_logName +
            SYSBMT_0000_experiment + SYSBMT_0000_LogExt ))
        or
            ( SYSBMT_0000_experiment = ':' );
    if SYSBMT_0000_experiment = ':' then
    SYSBMT_0000_error ('**** Experiment number > 9 is not alloved.');

{ Open new DOC file : }
SYSBMT_0000_DocNm := SYSBMT_0000_DocName +
                     SYSBMT_0000_experiment;
    assign  ( SYSBMT_0000_DocFile,
            SYSBMT_0000_DocNm + SYSBMT_0000_DocExt );
    rewrite ( SYSBMT_0000_DocFile );

{ Open new LOG file : }
SYSBMT_0000_LogNm := SYSBMT_0000_LogName +
                     SYSBMT_0000_experiment;
    assign  ( SYSBMT_0000_LogFile,
            SYSBMT_0000_LogNm + SYSBMT_0000_LogExt );
    rewrite ( SYSBMT_0000_LogFile );

SYSBMT_0000_CompFlag := SYSBMT_0000_exist
                        (SYSBMT_0000_DatName +
                        SYSBMT_0000_DatExt );

    if SYSBMT_0000_CompFlag then { If data file already exists }
                                 { create temporary file for checking }
        assign  ( SYSBMT_0000_DatFile,
                SYSBMT_0000_DatName + SYSBMT_0000_TempExt )
    else                         { create sample data file }
        assign  ( SYSBMT_0000_DatFile,
                SYSBMT_0000_DatName + SYSBMT_0000_DatExt );
        rewrite ( SYSBMT_0000_DatFile );
end; {_____ SYSBMT_0000_OpenFiles ___}

procedure SYSBMT_0000_Info ( var f : text );
{ General information }
begin

{ 1. Print title of this program : }
writeln ( f, SYSBMT_0000_title );
```

```
{ 2. Print the number of current experiment : }
writeln ( f, ' Experiment # ', SYSBMT_0000_experiment );

{ 3. Print date & time : }
writeln ( f, ' Execution:   date is ',
SYSBMT_0000_Day, '-', SYSBMT_0000_Month, '-', SYSBMT_0000_Year);
writeln ( f, '              time is ',
SYSBMT_0000_Hour, ':', SYSBMT_0000_Min, ':', SYSBMT_0000_Sec );

{ 4. Print file names : }
writeln ( f, ' Doc-file: ' , SYSBMT_0000_DocNm, SYSBMT_0000_DocExt,
          ' Log. file: ', SYSBMT_0000_LogNm, SYSBMT_0000_LogExt );

{ 5. Print final parameter of test loop : }
writeln ( f, ' Test loop: final variable is ',SYSBMT_0000_TestLoopFinal );

end; {_____ SYSBMT_0000_Info ___}

procedure SYSBMT_0000_Environment ( var f : text );
{ Environment information }
begin

{ 1. User information : }
writeln ( f, SYSBMT_0000_EnStr );

{ 2. What is version of compiler ? }
{$IFDEF VER40} writeln ( f, ' Turbo Pascal v. 4.0 '); {$ENDIF}
{$IFDEF VER50} writeln ( f, ' Turbo Pascal v. 5.0 '); {$ENDIF}
{$IFDEF VER55} writeln ( f, ' Turbo Pascal v. 5.5 '); {$ENDIF}
{$IFDEF VER60} writeln ( f, ' Turbo Pascal v. 6.0 '); {$ENDIF}

{ 3. What options are set ? }
write ( f, ' Options: ');
{$IFOPT A+} write ( f, 'A+'); {$ENDIF}
{$IFOPT B+} write ( f, 'B+'); {$ENDIF}
{$IFOPT E+} write ( f, 'E+'); {$ENDIF}
{$IFOPT I+} write ( f, 'I+'); {$ENDIF}
{$IFOPT N+} write ( f, 'N+'); {$ENDIF}
{$IFOPT R+} write ( f, 'R+'); {$ENDIF}
{$IFOPT S+} write ( f, 'S+'); {$ENDIF}
{$IFOPT V+} write ( f, 'V+'); {$ENDIF}
writeln ( f );

{ 4. What file is executed ? }
writeln ( f,' This file is ', SYSBMT_0000_ProgPath );

{ 5. What is DOS version ? }
writeln ( f, ' DOS version ', Lo ( SYSBMT_0000_ver ), '.',
                             Hi ( SYSBMT_0000_ver ) );

{ 6. Coprocessor detection : }
case Test8087 of
     0: write ( f,' No coprocessor');
     1: write ( f,' 8087 coprocessor');
     2: write ( f,' 80287 coprocessor');
     3: write ( f,' 80387 coprocessor');
end;
writeln ( f,' detected.');

end; {_____ SYSBMT_0000_Environment _____}

procedure SYSBMT_0000_TimeOut ( n : integer; var f : text );
{ Output run time of test }
var
     dt : longint;
begin

     GotoXY ( 1, WhereY );
     ClrEol;
     dt := SYSBMT_0000_TimeStop - SYSBMT_0000_TimeStart;
     writeln (' Test # ', n, ', time ', dt, ' ticks = ', dt/18.2:7:2, ' sec');

if ( dt < SYSBMT_0000_MinTime ) or ( dt > SYSBMT_0000_MaxTime )
     then begin                                    { error }
     writeln (
     '*** TimeOut ERROR: change SYSBMT_0000_TestLoopFinal const to ',
     SYSBMT_0000_AvrTime/dt * SYSBMT_0000_TestLoopFinal :10:0);

     if not SYSBMT_0000_NZeroFlag and ( dt < SYSBMT_0000_MinTime )
          then halt (1);
     end
     else                                          { ok }
     begin
          if SYSBMT_0000_DataFormat { = SYSBMT_0000_horizontal } then
               write (f, dt : 8 )
          else
               writeln (f, n:8, dt:8 );
     end;
end; {_____ SYSBMT_0000_TimeOut _____}

procedure SYSBMT_0000_TestResults;
{ Check data results }
var
     i : integer;
     l1, l2 : string;
begin
     write ( 'Test results : ');
     write ( SYSBMT_0000_LogFile, 'Test results : ' );
     Close ( SYSBMT_0000_DatFile );
     if SYSBMT_0000_CompFlag then                  { Checking is possible }
     begin
          assign     ( SYSBMT_0000_SampleFile,
                       SYSBMT_0000_DatName +
                       SYSBMT_0000_DatExt );       { Open sample data files }
          {$I-} reset ( SYSBMT_0000_SampleFile ); {$I+}
          reset ( SYSBMT_0000_DatFile );           { Reset current data file }
          i := 0;

{ Comparison of sample and current data files : }
          while not eof ( SYSBMT_0000_DatFile ) and
               not eof ( SYSBMT_0000_SampleFile ) do
               begin
                    inc ( i );
                    readln ( SYSBMT_0000_DatFile, l1 );
                    readln ( SYSBMT_0000_SampleFile, l2 );
                    if l1 < > l2 then              { files are different }
                    begin
                         writeln ( '*** Test result ERROR: line # ',i );
                         writeln ( l1 );
                         writeln ( l2 );
                         halt ( 2 );
                    end;
          end; { while }
          if eof ( SYSBMT_0000_DatFile )
          and
          eof ( SYSBMT_0000_SampleFile ) then { files compare Ok ! }
          begin
               close ( SYSBMT_0000_DatFile );
               close ( SYSBMT_0000_SampleFile );
               Erase ( SYSBMT_0000_DatFile );
               writeln ( ' Ok!' );
               writeln ( SYSBMT_0000_LogFile, ' Ok!' );
          end
          else                                     { files are different size }
          begin
               close ( SYSBMT_0000_DatFile );
               close ( SYSBMT_0000_SampleFile );
               SYSBMT_0000_error
               ( '*** Test result ERROR: different size of result files.' );
          end;
     end { if SYSBMT_0000_CompFlag ... }
     else { SYSBMT_0000_CompFlag = false : checking is impossible }
     begin
          writeln ( ' 1st DatFile: test abort.' );
          writeln ( SYSBMT_0000_LogFile,' 1st DatFile: test aborted.' );
     end;
end; {_____ SYSBMT_0000_TestResults___}

procedure SYSBMT_0000_List ( var FromF, ToF : text;
                             AutoexecFlag : Boolean );
var
```

Листинг 5 (продолжение)

```
    s : string;
    flag : Boolean;

begin
    flag := true;

    while not eof ( FromF ) do
    begin
        readln ( FromF, s );
        if pos ('{.+}', s ) > 0 then                { listing on }
            flag := true
        else
            if pos ('{.-}', s ) > 0 then            { listing off }
            begin
                flag := false;
                writeln ( ToF, '..... listing off .....');
            end;
        if flag then
            writeln ( ToF,  s );
    end;

    if AutoexecFlag then
        if ( SYSBMT_0000_ProgName < > '' ) then
        begin
            writeln ( ToF, 'cd ', SYSBMT_0000_ProgDir );
            writeln ( ToF, SYSBMT_0000_ProgName );
        end
        else
            writeln ( ToF, 'BMTEST.EXE' );

    writeln ( ToF );
    Flush ( ToF );
    close ( ToF );
    close ( FromF );

end; { _____ SYSBMT_0000_List _____}

procedure SYSBMT_0000_CopyF ( FromName, ToName : PathStr;
                              AutoexecFlag : Boolean);
var
    FromF, ToF : text;
begin

{ 1. Open 'From' file : }
assign ( FromF, FromName );
{$I-} reset ( FromF ); {$I+}
if IOResult < > 0 then
    SYSBMT_0000_error (' *** Error : file ' +FromName + ' not found ');

{ 2. Open 'To' file : }
assign ( ToF, ToName );
{$I-} rewrite ( ToF ); {$I+}
if IOResult < > 0 then
    SYSBMT_0000_error (' *** Error : file ' +ToName + ' not open ');

{ 3. Copy FromF ToF : }
    SYSBMT_0000_List ( FromF, ToF, AutoexecFlag );
end; {_____ SYSBMT_0000_CopyF _____}

procedure SYSBMT_0000_Listing;
var
    SourceFile, ListingFile : text;

begin
    assign ( SourceFile, SYSBMT_0000_ProgDir+
        SYSBMT_0000_ProgName + '.PAS' );
    {$I-} reset ( SourceFile ); {$I+}
    if IOResult = 0 then
    begin
        assign ( ListingFile, SYSBMT_0000_ListingNm );
        {$I-} rewrite ( ListingFile ); {$I+}
        if IOResult < > 0 then
            SYSBMT_0000_error ('*** ERROR : Can"t open listing file.' );
        SYSBMT_0000_List ( SourceFile, ListingFile, False );
    end
    else
        writeln ( '*** WARNING : source file ', SYSBMT_0000_ProgDir,
            SYSBMT_0000_ProgName, '.PAS not found !');
end; {_____ SYSBMT_0000_Listing ____}

procedure SYSBMT_0000_help;
var
    HelpFile : text;
begin
    assign ( HelpFile, SYSBMT_0000_HelpName );
    {$I-} reset ( HelpFile ); {$I+}
    if IOResult = 0 then
    begin
        clrscr;
        SYSBMT_0000_list( HelpFile, Output, False );
    end
    else
    SYSBMT_0000_error ( '*** ERROR : help file not found' );
end; {_____ SYSBMT_0000_help _____}

procedure SYSBMT_0000_GetParms;

procedure GetFirst ( s : string );

function ps ( ch : char ) : Boolean;
var
    p: integer;
    r: Boolean;
begin
    p: = pos ( ch, s );
    r: = p < > 0;
    if r then
        delete ( s, p, 1 );
        ps := r;
end; { _____ ps _____}

begin { GetFirst body }
SYSBMT_0000_ListFlag: =         ps ( SYSBMT_0000_ListO );
SYSBMT_0000_NZeroFlag : = not ps ( SYSBMT_0000_ZeroO );
SYSBMT_0000_HelpFlag: =         ps ( SYSBMT_0000_HelpO );
if s < > ''
    then
        SYSBMT_0000_error ( '*** ERROR : invalid parameter ' + s );
end; {_____ GetFirst _____}

begin { SYSBMT_0000_GetParms body }
    if ParamCount = 0 then
        GetFirst ( SYSBMT_0000_DefaultParm )
    else
        GetFirst ( ParamStr (1) );

    if SYSBMT_0000_HelpFlag then
        SYSBMT_0000_Help;

    ifParamCount > 1 then
        SYSBMT_0000_ListingNm : = ParamStr ( 2 )
    else
        SYSBMT_0000_ListingNm : = SYSBMT_0000_DefListingNm;
end; {_____ SYSBMT_0000_GetParms _____}

procedure SYSBMT_0000_ReBoot;
{ DOS reboot }
var
    reset : word absolute $0040 : $0072;
    JmpBIOS : procedure;
begin
    reset : = $1234;
    @JmpBIOS: = Ptr ( $FFFF, $0000 );
    JmpBIOS;
end; {_____ SYSBMT_0000_ReBoot _____}

procedure SYSBMT_0000_ChConfig;
{ Change CONFIG.SYS & AUTOEXEC.BAT }
var
    CompPath, driver : PathStr;
```

```
    ch : char;
    SysFile : text;

begin { SYSBMT_0000_ChConfig body }
    writeln ( '*** WARNING: special AUTOEXEC.BAT is not installed !');
    write ( ' Change CONFIG.SYS & AUTOEXEC.BAT & DOS reboot (Y/N) ?' );
    SYSBMT_0000_beep;

    repeat
        ch := UpCase ( ReadKey );
    until ch in ['Y', 'N'];
    writeln ( ch );

    if ch = 'N' then
        exit;

{ Change : }

{- find system driver }
    CompPath := GetEnv ( 'COMSPEC' );
    driver := Copy ( CompPath, 1, 2 ) + '\';

{ - test }
    if SYSBMT_0000_exist ( driver + 'AUTOEXEC.B-M' )
    or
    SYSBMT_0000_exist ( driver + 'CONFIG.B-M' ) then
        SYSBMT_0000_error
        ('*** ERROR : ' + driver + ' .B-M files already exist.');

{ - change AUTOEXEC.BAT }
    assign ( SysFile, driver + 'AUTOEXEC.BAT' );
    rename ( SysFile, driver + 'AUTOEXEC.B-M' );
    SYSBMT_0000_CopyF  ('AUTOEXEC.BMT', driver +
                        'AUTOEXEC.BAT', True );

{ - change CONFIG.SYS }
    assign ( SysFile, driver + 'CONFIG.SYS' );
    rename ( SysFile, driver + 'CONFIG.B-M' );
    SYSBMT_0000_CopyF  ('CONFIG.BMT', driver +
                        'CONFIG.SYS', False );

SYSBMT_0000_ReBoot;                { DOS reboot       }
end; {_____________________ SYSBMT_0000_ChConfig _____________}

procedure SYSBMT_0000_Start;

begin
    ClrScr;
    NoSound;
    SYSBMT_0000_ver := DosVersion;
    if Lo ( SYSBMT_0000_ver ) > 2 then
    begin
        SYSBMT_0000_ProgPath := ParamStr ( 0 );
        FSplit ( SYSBMT_0000_ProgPath, SYSBMT_0000_ProgDir,
            SYSBMT_0000_ProgName, SYSBMT_0000_ProgExt );
    end
    else
        SYSBMT_0000_ProgName := '';

    SYSBMT_0000_EnStr := GetEnv ( SYSBMT_0000_EnvNm );
    if ( SYSBMT_0000_EnStr = '' )
    and ( SYSBMT_0000_ProgName < > SYSBMT_0000_CompilerNm )
    then begin
        writeln ( SYSBMT_0000_title );
        SYSBMT_0000_EnStr := 'Environment var. '+
                             SYSBMT_0000_EnvNm + ' was not set.';
        SYSBMT_0000_ChConfig;
    end;

    SYSBMT_0000_GetParms;
    SYSBMT_0000_OpenFiles;
    GetTime ( SYSBMT_0000_Hour, SYSBMT_0000_Min,
              SYSBMT_0000_Sec, SYSBMT_0000_Sec100 );
    GetDate ( SYSBMT_0000_Year, SYSBMT_0000_Month,
              SYSBMT_0000_Day,SYSBMT_0000_DayOfWeek );
    SYSBMT_0000_Info ( Output);
    SYSBMT_0000_Info ( SYSBMT_0000_LogFile );
    SYSBMT_0000_Environment ( Output );
    SYSBMT_0000_Environment ( SYSBMT_0000_LogFile );
end; {_____________________ SYSBMT_0000_Start _____________}

procedure SYSBMT_0000_Finish;

var
    driver, CompPath : PathStr;
    ch : char;

procedure Reduction ( s1 : PathStr; s2 : ExtStr );
{ Reduct the old AUTOEXEC.BAT & CONFIG.SYS files }
var
    f : text;
begin
    assign ( f, s1 + s2 );
    erase( f );

    assign ( f, s1 + '.B-M' );
    rename ( f, s1 + s2 );

end; {_____________________ Reduction _____________________}

procedure quit;
var
    ch : char;
begin
    writeln ( 'Press any key for quit...');
    SYSBMT_0000_beep;
    ch := ReadKey;
end; {_____________________ quit _____________________}

begin
    SYSBMT_0000_TestResults;
    writeln ( SYSBMT_0000_LogFile, ' Tests are completed.');
    close ( SYSBMT_0000_LogFile );
    close ( SYSBMT_0000_DocFile );
    if SYSBMT_0000_ListFlag
    and ( SYSBMT_0000_ProgName < > SYSBMT_0000_CompilerNm ) then
        SYSBMT_0000_Listing;
    writeln ( '= = = = = The end of program. = = = = = = = = = = = = = = = =');

    if SYSBMT_0000_EnStr = '' then
        quit
    else                                        { change config? }
    begin
    {- find system driver }
        CompPath := GetEnv ( 'COMSPEC' );
        driver := Copy ( CompPath, 1, 2 ) + '\';

    { - test }
    if SYSBMT_0000_exist ( driver + 'AUTOEXEC.B-M' )
    and
    SYSBMT_0000_exist ( driver + 'CONFIG.B-M' ) then
    begin
        SYSBMT_0000_beep;
        write('Restore the old AUTOEXEC.BAT & CONFIG.SYS files (Y/N) ?');
        ch := UpCase ( ReadKey );

        if ch = 'Y' then
        begin
            Reduction ( driver + 'AUTOEXEC', '.BAT' );
            Reduction ( driver + 'CONFIG','.SYS');
            SYSBMT_0000_Reboot;
        end
        else
            write ('N');
        end
        else
            quit;
    end;

end; {_____________________ SYSBMT_0000_Finish _____________}
```

Профессиональная работа в среде Microsoft QuickBASIC

А. Колесов, О. Павлова

Часть 1. Технология разработки программ

Несмотря на бытующее мнение о программировании в QuickBASIC как о некотором очень простом деле, освоение всех возможностей системы требует от разработчика определенных усилий и знаний. Одной из главных причин этого является то, что QuickBASIC представляет собой симбиоз интерпретирующих и компилирующих систем программирования. Программистам, имеющим опыт работы с компиляторами, приходится осваивать специфическую технологию отладки программ в среде интерпретатора, а программисты, пришедшие из традиционного интерпретатора BASIC, впервые сталкиваются с такими понятиями, как "модуль", "процедура", "структурное программирование" и так далее.

В предлагаемой Вашему вниманию статье авторы делают попытку рассмотреть основные вопросы технологии программирования в системе QuickBASIC на примере версии 4.5. Все сказанное здесь будет верно и для последующих версий Microsoft BASIC 7.0 и 7.1, но читатели должны иметь в виду, что там появилось много новых возможностей.

Структура BASIC программы

BASIC программа состоит из главной части, внешних процедур и некоторых других самостоятельных программных блоков. Программные блоки реализуются в виде исходных файлов — модулей программы, которые могут раздельно компилироваться, автономно загружаться и выгружаться интерпретирующей системой, объединяться в библиотеки и прочее (рис. 1).

Данное определение программы отражает общую логику и показывает два основных аспекта технологии разработки программы:

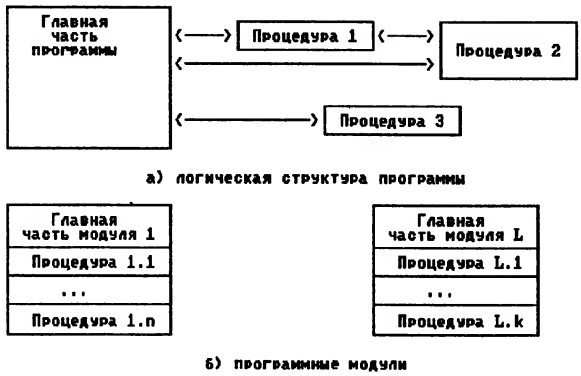


Рис. 1. Схема организации программы

- структурную организацию программы на алгоритмическом уровне из логических блоков;
- конкретную реализацию этих логических блоков в виде автономных программных модулей.

Следует внести ясность в термины "процедура" и "модуль". В классических компиляторах, например, в Фортране, один модуль содержал только одну процедуру, поэтому эти понятия были практически синонимами. В QuickBASIC, как и в других современных языках, программный модуль имеет гораздо более сложную структуру.

Модуль BASIC программы состоит из главной части (в документации Microsoft она называется "module-level code") и процедур: одной, нескольких, а может быть, и ни одной. Текст главной части также может отсутствовать, хотя редактор среды QuickBASIC выделяет для него окно.

Такая организация программных модулей предоставляет программисту много дополнительных возможностей, но в то же время создает некоторые новые

проблемы. Например, как следует проводить разбиение (или объединение) процедур по нескольким модулям?

Естественное решение на начальном этапе разработки программы — размещение всех процедур в одном модуле. Но с ростом программы появляется необходимость перехода к многомодульной организации. И тогда программист сталкивается с тем, что просто взять и "разрезать" модуль на составные части можно далеко не всегда: некоторые процедуры оказываются довольно крепко связаны вместе (например, общими переменными).

Проблема выделения модулей и процедур программы будет обсуждаться во второй части статьи, а сейчас необходимо только уточнить приведенное определение программы: программа состоит из главной части и внешних процедур, которые могут быть объединены в одну или несколько логических групп на уровне программного модуля, где модуль — это совокупность взаимосвязанных процедур программы (общие данные, программные блоки и прочее).

Форматы программных модулей

Программные модули, из которых создается прикладная программа, в общем случае состоят из двух характерных групп файлов: прикладные модули, которые разрабатываются в ходе реализации данной задачи, и модули инструментальных средств, сделанные ранее и используемые как готовые конструкции.

Все программные модули могут быть представлены в двух принципиально разных форматах кодирования — исходном (BAS) и объектном (OBJ или LIB). Исходные модули содержат тексты программ, объектные — специальные двоичные коды, полученные в результате обработки исходных модулей компилятором. Файлы объектных библиотек LIB получаются в результате объединения нескольких объектных модулей и ранее созданных программой LIB.EXE библиотек.

Корректировать можно только исходные модули. Поэтому прикладные модули могут быть представлены только в исходном формате. Модули инструментальных средств могут быть представлены как в исходном, так и в объектном форматах. Процедуры, написанные на других языках программирования, используются только в виде объектных модулей.

Исходные модули могут, в свою очередь, храниться на диске в двух форматах — Text и QuickBASIC. Text — это обычный символьный (ASCII) формат данных, который может обрабатываться любым текстовым редактором. В оперативной памяти в среде QB.EXE исходные модули хранятся в собственном формате QuickBASIC, который более удобен для работы интерпретатора.

Мы рекомендуем использовать формат Text: компилятор BC.EXE надежно работает только с ним, хотя операции SAVE/LOAD выполняются несколько медленнее. Кроме того, Microsoft BASIC 7.0 сможет прочитать только формат Text (в нем используется другой внутренний формат данных — Binary).

Имейте в виду, что при создании нового модуля в среде QB.EXE по умолчанию предполагается использование формата QuickBASIC (в QBX.EXE версии 7.0 — формата Text). Для перехода в текстовый формат надо воспользоваться командой SAVE AS, в дальнейшем установленный таким образом формат будет поддерживаться автоматически.

Общая схема разработки программ

Разработка программ в системе QuickBASIC включает два основных режима: работа в среде интерпретатора QB.EXE и создание автономного EXE модуля программы (рис. 2). На первый этап приходится основной объем работы — написание и отладка исходных текстов программы.

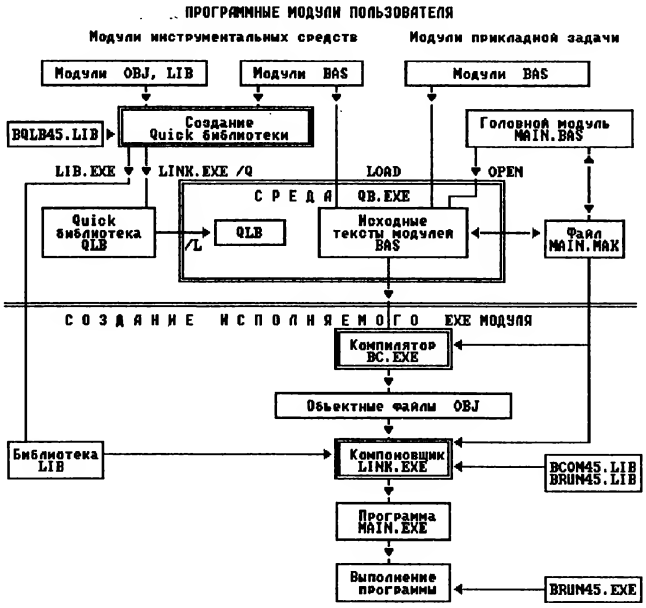


Рис. 2. Общая схема разработки программ в системе QuickBASIC 4.5

Выполнение готовой программы возможно как в среде QB.EXE, так и в виде автономного EXE модуля. Использование EXE модуля является наиболее эффективной формой рабочей эксплуатации программ. Однако бывают ситуации, когда удобнее выполнять программу в среде. Например, изучение библиотек подпрограмм с помощью демонстрационных программ, написанных на языке BASIC, или выполнение оперативных расчетов по многочисленным программам-однодневкам.

Создавать EXE модули, объектные и Quick библиотеки можно как по командам из среды QB.EXE, так и автономно. При этом работа в среде, конечно, более удобна и проста, зато при автономной работе с программами BC.EXE, LIB.EXE, LINK.EXE пользователь получает больше возможностей для управления режимами их работы.

Использование библиотек подпрограмм

Классическая схема BASIC интерпретатора не позволяла работать с двоичными библиотеками подпрограмм, что являлось его серьезным недостатком. В QuickBASIC впервые реализована возможность применения библиотек на уровне интерпретатора. Поэтому пользователь этой системы имеет дело с библиотеками двух форматов — обычной объектной библиотекой (LIB) и Quick библиотекой (QLB).

По существу Quick библиотека представляет собой выполняемый файл специального формата, который загружается одновременно с QB.EXE и является как бы функциональным расширением среды интерпретатора.

Кроме этих двух типов библиотек, в QuickBASIC реализован еще один новый формат библиотечного файла поддержки — Run-Time Module. Он является своеобразным аналогом Quick библиотеки на уровне EXE модуля. В версии 4.5 используется только один файл поддержки — BRUN45.EXE, но в версии 7.0 уже имеется целый набор таких модулей. Кроме того, для пользователя там появилась возможность создавать свои собственные файлы поддержки.

Работа в среде QB.EXE

Одним из основных принципов работы в среде QB.EXE является то, что для запуска программы на выполнение в оперативную память должны быть загружены все программные модули (процедуры), необходимые для ее функционирования. Именно такая организация работы обеспечивает предельно высокую скорость отладки программы, сводя к минимуму временные затраты на ее перетрансляцию и компоновку.

Соответственно в QB.EXE есть ограничение на размеры программы и массивы данных. Реальный объем загруженных исходных модулей не может превышать 5000 — 5500 строк. Решить эту проблему можно достаточно просто, используя Quick библиотеки. С другой стороны, вопрос создания больших программных комплексов должен решаться не за счет увеличения размеров одной программы, а путем реализации оверлейных структур EXE модулей и создания многомодульных комплексов. Но это — тема отдельного разговора.

Отметим следующие основные моменты работы в среде QB.EXE.

1. Работа с программой в среде начинается с загрузки исходного текста ее главного модуля (например, MAIN.BAS — рис.2) в оперативную память командой OPEN или с его создания командой NEW меню File. Все остальные процедуры, необходимые для выполнения программы, могут быть загружены либо в виде текстов исходных модулей, которые могут подвергаться коррекции и отладке, либо в виде Quick библиотек.

2. Прикладные модули программы должны загружаться в виде исходных модулей, так как они являются объектом разработки и отладки.

3. Все исходные модули могут загружаться в оперативную память и выгружаться из нее в любой момент работы командами LOAD и UNLOAD. Можно создавать новые программные модули командой CREATE.

4. В процессе одного сеанса работы в среде можно использовать только одну Quick библиотеку. Она загружается в оперативную память одновременно при запуске QB.EXE с ключом /L:

```
QB.EXE /L [имя Quick библиотеки].
```

По умолчанию загружается библиотека QB.QLB, входящая в состав системы QuickBASIC.

5. Процедуры, написанные на других языках программирования, можно использовать только с помощью Quick библиотек.

6. В ходе работы QB.EXE постоянно отслеживает состав всех загруженных в оперативную память исходных модулей в специальном текстовом файле, имя которого состоит из имени главного модуля программы плюс расширение MAK. Этот файл используется для последующих операций загрузки модулей программы, а именно при выполнении операции OPEN производится поиск соответствующего файла MAK и затем автоматическая загрузка всех исходных модулей, записанных в нем.

Создание выполняемого EXE модуля

Создание выполняемого файла производится, как правило, после завершения процесса разработки и отладки программы. При этом последовательно выполняются операции компиляции (BC.EXE) и компоновки (LINK.EXE).

Формирование EXE модуля может производиться при работе в среде QB.EXE или автономно, вне среды. При запуске команды Make EXE File из меню Run система автоматически компилирует все исходные модули программы, список которых определяется текущим MAK файлом, и компонует их в EXE модуль. Если при запуске среды была указана загрузка Quick библиотеки, то программа LINK.EXE подключает объектную библиотеку LIB с таким же именем.

В версии 4.5 реализована возможность включения средств отладки, а также имеются два основных варианта создания EXE модуля — "Stand-Alone" и "Requiring BRUN45.EXE". В первом случае получается автономный загрузочный модуль программы, а во втором — более компактный модуль (в среднем на 40-50 Кбайтов меньше), но требующий при выполнении модуля поддержки BRUN45.EXE, дозагрузка которого производится автоматически. Возможности создания EXE модулей в версии 7.0 получили очень существенное развитие, и всеми этими режимами можно управлять непосредственно из среды QBX.EXE.

Внимание: могут возникнуть проблемы

В системе QuickBASIC имеются два транслятора — интерпретатор QB.EXE и компилятор BC.EXE. Хотя, в принципе, они должны обладать одинаковыми функциональными возможностями, все-таки это две разные программы, которые могут в чем-то отличаться друг от друга (на самом деле так оно и есть). Поэтому пользователь должен быть психологически готов, что программа, прекрасно работающая в среде QB.EXE, начинает выдавать сообщения об ошибках на этапе создания EXE модуля или неверно работать при выполнении.

В качестве примера мы приведем несколько наиболее характерных ситуаций, обнаруженных нами в версии 4.5.

1. Начнем с самой известной проблемы — EXE модули не хотят вводить строчную русскую букву "р". Известно несколько вариантов решения этого вопроса в версии 4.5, а в компилирующей системе версии 7.0 эта ошибка устранена.

2. BC.EXE не может корректно обрабатывать некоторые операторы исходных модулей, записанные в формате QuickBASIC, что, в принципе, должен бы делать. Данная ситуация может возникнуть как при компиляции, так и при компоновке программ. Поэтому, если на этапе создания EXE модуля неожиданно появятся странные сообщения об ошибках, в первую очередь проверьте формат исходных файлов программы и переведите его в формат Text.

3. При использовании конструкций с применением символьных выражений типа:

```
a$ = a1$ + a2$+ ... aN$
```

следует иметь в виду, что QB.EXE может обрабатывать оператор с любой суммарной длиной символьных переменных, у BC.EXE возможности такой обработки ограничены. Поэтому он может выдать сообщение об ошибке: "Expression too complex". В этом случае надо разделить один оператор на несколько:

```
a$ = a1$ + a2$ + a3$
a$ = a$ + a4$ + ...
...
```

4. При переопределении размерности динамического массива QB.EXE автоматически устанавливает его тип в соответствии с первоначальным определением, а компилятор BC.EXE фиксирует ошибку при несовпадении типов данных (изменение типа переменных динамических массивов не разрешается, можно корректировать только их размерность). Это видно из примера 1.

5. В среде QB.EXE последовательность операторов

```
A = 0: B = 0: C = A / B
```

будет успешно выполнена, значение C будет равно нулю. А EXE модуль выдаст сообщение об ошибке при делении на ноль, и программа аварийно завершится.

6. При работе с EXE модулем необходимо учитывать, что по умолчанию программа BC.EXE средства отладчика не включает, в то время как интерпретатор всегда использует многие контролирующие функции отладчика. В частности, при выходе значения индекса массива за его границы QB.EXE выдает сообщение об ошибке, а работа EXE модуля может привести к зависанию.

7. При работе с EXE модулями отмечен ряд ошибок, имеющих случайный характер и появляющихся крайне редко. Например, конструкция

```
FOR k = 1 TO Km: ... : NEXT
```

при значении Km=0 обычно не допускает попадания внутрь цикла, но иногда это происходит, поэтому лучше выполнить проверку величины Km перед обращением к циклу.

Другой пример подобной ситуации — передача значений параметров с помощью арифметического выражения при обращении к оператору или процедуре, например:

```
PAINT (x+1,y+1),ForeGround%,15
```

В некоторых случаях такое обращение вызывает искажение переменной ForeGround% со всеми вытекающими последствиями. А конструкция

```
x1=x+1: y1=y+1
PAINT (x1,y1),ForeGround%,15
```

работает надежно всегда. Мы не призываем читателей отказаться от использования арифметических выражений при передаче параметров (ошибки возникают крайне редко). Просто при появлении каких-то непонятных ошибок в работе программы следует обратить особое внимание на подобные конструкции.

Справедливости ради надо отметить, что все сказанное совсем не означает, что качество компилятора ниже, чем интерпретатора. Ведь никто не проверял

Пример 1

```
' первоначальное определение динамического массива
REDIM Array(10) AS STRING
Array(1) = "тип — символьный, переменной длины"
' переопределение длины массива
REDIM Array(20) AS STRING  ' здесь все правильно
'
REDIM Array(20)            ' для QB.EXE — это повторение предыдущей
'                          ' строки, а BC.EXE выдаст ошибку:
                           ' "AS clause required"
'
REDIM Array(20) AS INTEGER     ' здесь явная ошибка,
                           ' QB.EXE сообщит:
                           ' "Duplicate definition"
```

этого, делая обратный переход из компилятора в интерпретатор.

Создание библиотек

Создание библиотек из модулей, написанных на языке QuickBASIC, наиболее удобно выполнять в среде QuickBASIC. Для этого все модули, которые войдут в новую библиотеку, следует загрузить в среду QB.EXE (как и при создании программы) в виде исходных модулей или в виде ранее созданной Quick библиотеки. По команде Make Library из меню Run последовательно формируются объектная и Quick библиотеки.

На практике формирование библиотек довольно часто приходится выполнять путем объединения ранее полученных объектных модулей или библиотек (например, при использовании процедур, написанных на других языках). Такая работа может проводиться только вне среды QuickBASIC.

Особенностью работы в среде QB.EXE является то, что интерпретатор может работать только с одной Quick библиотекой. Поэтому наиболее удобным представляется следующий порядок работы с библиотеками подпрограмм.

Предположим, что имеется несколько библиотек подпрограмм, записанных в объектом формате, — LIB1, LIB2, ... В ходе разработки прикладной программы необходимо использовать некоторые из этих библиотек, например, LIB1 и LIB2. Для этого надо произвести следующие операции:

а) создать объектную библиотеку LIBMAIN.LIB:

```
LIB.EXE LIBMAIN.LIB+LIB1.LIB+LIB2.LIB;
```

б) создать Quick библиотеку LIBMAIN.QLB:

```
LINK.EXE /Q
LIBMAIN.LIB,LIBMAIN.QLB,,BQLB45.LIB;
(BQLB45.LIB — специальная библиотека
              системы QuickBASIC);
```

в) запустить среду:

```
QB.EXE /L LIBMAIN ■
```


Рекомендуемая литература

1. Microsoft QuickBASIC for IBM personal computers and compatibles. — Microsoft Corporation, 1987.
2. В.П.Дьяконов. От простого Бейсика к быстрому// Мир ПК. - 1991. - № 4.
3. А.А.Колесов. QuickBASIC — это то, что Вам нужно//КомпьютерПресс. - 1991.- № 3.
4. А.А.Колесов, О.Р.Павлова. Все течет, все изменяется, или от "быстрого" Бейсика к "профессиональному"// Мир ПК. - 1992. - № 5.

Информационное сообщение

В декабре 1992г. в Москве по инициативе сотрудников института инженерных изысканий в строительстве (ПНИИИС) была начата работа по созданию Ассоциации пользователей языка Microsoft BASIC (АПМБ). Предполагалось, что эта организация будет объединять пользователей всей линии языка Microsoft BASIC — GW, Quick, Professional, Visual.

Цель подготовительного этапа состояла, прежде всего, в сборе информации о пользователях языка: их составе, интересах, желании участвовать в работе Ассоциации и прочем. К началу марта были установлены двусторонние связи с более чем 100 пользователями, примерно половина которых находится вне Москвы и представляет практически все регионы СНГ. Кроме того, были налажены рабочие контакты с российским представительством фирмы Microsoft в Москве и их дистрибьютерами, изучался опыт работы других групп пользователей, в том числе по линии Международного Компьютерного Клуба.

18 февраля в ПНИИИС состоялось организационное собрание группы пользователей г. Москвы, на котором присутствовало около 50 человек. Обсуждались различные аспекты проблемы создания АПМБ, также были определены следующие основные задачи:

- обеспечение контактов между самими программистами, а также с представительством Microsoft, другими группами пользователей;
- учебно-консультационная деятельность;
- координация деятельности региональных групп программистов;
- оказание помощи программистам в распространении их разработок в России и за рубежом;
- содействие в охране авторских прав на программные продукты;
- предоставление информации о продуктах фирмы Microsoft;
- организация финансирования компьютерных проектов.

На этом собрании было принято принципиальное решение о создании АПМБ. Но вопросы о ее конкретной организационной структуре, финансировании работы и так далее остаются на сегодня открытыми: пока вся работа ведется на общественных началах, за счет небольшой инициативной группы. Поэтому сейчас мы крайне заинтересованы в самой разнообразной поддержке нашей деятельности со стороны спонсоров. Тем более, что, несмотря на обещанную руководством Microsoft поддержку (информация об этом прозвучала даже в публичных выступлениях, в том числе и в печати), до настоящего времени никакого интереса к деятельности группы пользователей BASIC фирма не проявляла.

Тем не менее АПМБ уже ведется некоторая полезная работа:

- сбор и распространение информации о пользователях BASIC и их разработках;
- предоставление информации о продуктах Microsoft и оказание помощи в их приобретении;
- обмен опытом между разработчиками;
- оказание помощи в публикации материалов в компьютерных изданиях.

Мы приглашаем всех любителей языка BASIC принять участие в создании и работе АПМБ. При этом хотелось бы подчеркнуть, что успех ее деятельности в основном зависит от активной позиции каждого из нас. "Спасение утопающих — дело рук самих утопающих!"

Координатор АПМБ А.А.Колесов

Адрес для писем: 107061 Россия, Москва, Б.Черкизовская ул., 6-4-106.

Телефоны: (095) 366-25-02 раб., 161-23-17 дом. ■

Информация о новых вирусах в формате антивирусной программы AVSP 2.0.

AVSP предоставляет пользователю возможность непостредственного ввода информации о новых вирусах в программу. Благодаря регулярной публикации такой информации AVSP на Вашем компьютере не отстанет от времени и создателей новых вирусов.

AVSP содержит ряд средств борьбы с неизвестными вирусами. ДИАЛОГ-МГУ обращается ко всем читателям с просьбой присылать информацию об обнаруженых новых вирусах.

Для ввода приведенной ниже информации в Вашу программу войдите в пункт меню "Данные о файлах и вирусах", подменю "Изменение информации о вирусах". Для более подробного описания можно обратиться к документации (пункт "Ввод информации о вирусах из журналов").

Внимание !!! Новая информация по вирусу Leninformatika.

Имя вируса: Leninformatika
Номер: 189.2-49150
Тип вируса: PROGRAM
Размер 23693
Шаблон: 268E1E2C008BFE268B751683C71FAC
Процедуры: *.COM { Opt(End:-23581) Mov(MP:460,Beg:0,3) Tr(End:-23693) }
Справка: Рисует комьютер и пишет текст "Leninformatika" в графическом режиме. Похож на вирус Vienna.

Имя вируса: Loz-2968
Номер: 233.1-37097
Тип вируса: PROGRAM
Шаблон: 061EE800005E2E8A44F53C007411
Справка: Резидентный. Содержит рекламу программе AIDSTEST.

Имя вируса: SVC-6.01
Номер: 234.1-9849
Тип вируса: PROGRAM
Шаблон: 2EC6840712012EC6840812018CD7
Справка: См. описание вируса SVC-6. Содержит также текст "Moscow SVC 6.01".

Имя вируса: SVC-6 Boot
Номер: 235.1-42935
Тип вируса: BOOT
Шаблон: B402B00ABB007EB90200BA8000
Справка: См. описание вируса SVC-6.

По вопросам приобретения AVSP 2.0 обращайтесь в ДИАЛОГ-МГУ по телефонам (095) 939-38-90, 939-38-91. Подробная информация высылается бесплатно.

Цена одного комплекта AVSP 2.0 на 15 декабря 1992 г. составляет 3848 рублей (с учетом 28% налога). При покупке нескольких комплектов предоставляется скидка. Скидка для дилеров - до 60%.

СУБД FoxPro 2.0 — новый лидер семейства xBase

М. Корнеев

Основные сведения

В середине 1991 года фирма Fox Software, широко известная у нас благодаря СУБД линии FoxBase, выпустила на рынок вторую версию своей новой линии СУБД — FoxPro. (Эта фирма была вскоре куплена Microsoft Corp.) Те, кто в свое время были приятно поражены преимуществами FoxPro 1.x, по сравнению с другими распространенными в то время системами семейства xBase (FoxBase, Clipper, dBase IV), были еще более приятно поражены тем, что предоставляла пользователям вторая версия системы.

Список всех новаций, появившихся в FoxPro 2.0, слишком велик, поэтому в статье будут кратко рассмотрены только основные новшества: расширенная версия для 386-го процессора, компилятор, средства создания меню, экранов и запросов, расширение возможностей при создании отчетов и этикеток, средства поддержки работы над проектом, новые форматы индексных файлов, новая технология поиска и обработки запросов, встроенные команды языка SQL, расширение возможностей команды READ, низкоуровневый интерфейс прикладных программ (API), драйвер принтера, функции работы с массивами.

Стоит также упомянуть о том, что вторая версия системы позволяет редактировать макросы, полностью поддерживает реляционные отношения типа один_ко_многим (это реализовано и в команде BROWSE, причем двояким образом), имеет контекстно чувствительную помощь с интересной опцией See Also (смотри также), причем менеджер помощи может применяться в пользовательских программах, и, наконец, имеет около 200 новых и усовершенствованных команд и функций.

Кроме того, FoxPro 2.0 занимает меньше памяти и, в принципе, работает быстрее первой версии на 10-15 процентов на том же наборе команд (более подробно о быстродействии будет сказано далее).

Технология Рашмора

Самые поразительные новшества FoxPro 2.0 связаны с использованием нового запатентованного фирмой способа поиска и обработки запросов, называемого технологией Рашмора (Rushmore Technology). Эта технология основана на более глубоком внутреннем использовании доступных индексов в многочисленных командах, использующих опцию FOR (в их числе теперь и команда BROWSE) и команды SET FILTER TO. Суть технологии Рашмора состоит в том, что доступ к базе данных при определенных условиях может быть подвергнут внутренней оптимизации, что позволяет в сотни и тысячи раз сократить время выполнения команд. Условие, достаточное для того чтобы запрос был оптимизирован, простое и легко выполнимое — опция FOR должна быть единственной ограничивающей опцией, а логическое выражение должно состоять из базовых оптимизируемых выражений, объединенных логическими операторами. Под базовыми оптимизируемыми выражениями понимаются выражения вида:

<индексное выражение> <оператор отношения> <константа>

или

<константа> <оператор отношения> <индексное выражение>.

Под индексным выражением понимается выражение, в точности соответствующее тому, по которому создан и доступен индекс или тэг (tag) мультииндексного файла, под константой — любое выражение типа константы, переменной или поля базы из другой области, а под оператором отношения — любой из операторов "=, <, >, <=, >=, #, !=, <>". Кроме того, выражение для индекса не должно содержать оператор "!", и использовать опции FOR и UNIQUE. Помимо оптимизируемых выражений, существуют и так называемые частично оптимизируемые выражения. Ес-

ли выражение неоптимизируемое или недостаточно свободной памяти, система обрабатывает запросы так, как это делают все остальные СУБД этого класса.

Заметьте, что названия индексов или тэгов в мультииндексных файлах не играют роли для оптимизации, значение имеет их доступность. С этим связано большое преимущество, которое получает пользователь, применяющий новый формат индексных файлов, — мультииндексный файл, аналогичный используемому в dBase IV, за исключением того, что он имеет меньшие размеры и может содержать неограниченное число тэгов. Другие преимущества связаны с тем, что открывается один файл вместо нескольких. Кроме того, система поддерживает так называемый структурный индексный файл, открывающийся автоматически при открытии DBF файла, и позволяет упорядочивать записи с использованием одного и того же индекса или тэга как по возрастанию, так и по убыванию.

Таким образом, если в DBF файле есть поля Модель, Процессор, Диск, Видео, Цена и доступны индексы и/или тэги по выражениям Процессор, Диск, UPPER(Видео), Цена, то оптимизируемая команда

```
BROWSE FOR (Процессор = "80386DX" AND ЦЕНА <= 1400 OR ;
            Процессор = "80386SX" AND ЦЕНА <= 1200) AND ;
            Диск >= 80 AND Диск <= 160 AND ;
            UPPER(Видео) = "SVGA"
```

выведет на экран таблицу за доли секунды. Причем количество записей в базе (300 или 1000000) не имеет существенного значения для времени выполнения. Команда с условием

```
Модель = "DeskPro" AND Цена < 3500
```

будет частично оптимизируемой и исполнится несколько медленнее, а с условием Видео = "VGA" — неоптимизируемой, с временем выполнения, прямо пропорциональным количеству записей (но все же меньшим, чем у Clipper, dBase IV или FoxPro 1.x).

Можно сказать, что чем больше записей в базе и чем меньше их должно оказаться в результирующей выборке, тем эффективнее будет использование технологии Рашмора по сравнению с обычными методами. Таким образом, FoxPro 2.0 дает возможность обрабатывать гигантские базы с миллионами записей в приемлемое для диалоговой системы время.

Предложения языка SQL

В язык FoxPro 2.0 включены 3 предложения языка SQL: CREATE TABLE | CURSOR, INSERT и SELECT.

Предложение определения данных CREATE TABLE | DBF позволяет создать пустой DBF файл (в терминологии SQL — таблицу) со специфицированной в предложении структурой. Реализация предложения CREATE CURSOR, строго говоря, не имеет отношения ни к SQL, ни к его курсорам, а просто создает временную таблицу, с которой можно манипулировать как с обычным DBF файлом и которая исчезнет после закрытия таблицы.

Предложение INSERT относится к классу предложений манипулирования данными SQL и позволяет добавлять записи в таблицу. INSERT открывает таблицу, если она не открыта, и добавляет запись со специфицированными в предложении значениями полей. INSERT работает несколько быстрее соответствующего набора команд xBase.

Безусловно, одним из важнейших новшеств языка Foxpro 2.0 является наличие в нем предложения SELECT. Реализация этого предложения манипулирования данными имеет в FoxPro некоторые отличия от стандарта SQL, заключающиеся в том, что FoxPro позволяет направить результат запроса в DBF файл или временную таблицу, с которыми можно манипулировать обычным для FoxPro образом. В остальном реализация этого предложения соответствует стандарту и включает почти все определенные в SQL стандартные функции, фразы, предикаты и кванторы. К сожалению, уровень подзапросов пока ограничен одним.

Мощь этого предложения определяется возможностью его внутренней оптимизации по технологии Рашмора. При наличии соответствующих индексов самые сложные запросы к нескольким реляционно связанным таблицам, насчитывающим сотни тысяч записей, выполняются в считанные секунды. Внутренний оптимизатор FoxPro, имея доступ к различной, в том числе и недоступной программисту информации, сам строит стратегию выполнения запроса. При этом учитывается наличие тех или иных индексов, размер файлов, объем доступной памяти и тому подобное. В случае отсутствия необходимых индексов FoxPro сам построит их. Одно предложение SELECT может заменить десятки строк xBase программы и, вообще говоря, в подавляющем большинстве случаев будет работать быстрее xBase программ, написанных самыми опытными программистами.

Следует иметь в виду, что в предложении SELECT вы можете обращаться и к неоткрытым таблицам. В этом случае FoxPro сам откроет их, но при этом ему будут доступны только тэги, имеющиеся в структурном индексном файле!

Фактически технология Рашмора меняет сложившийся в xBase стиль программирования работы с базой данных. Вряд ли кому-нибудь в последнее время приходило в голову употребить, например, команду SET FILTER TO по отношению к базе, состоящей из сотен тысяч записей, в программе, ведущей диалог с пользователем в реальном масштабе времени, а не печатающей по ночам какой-нибудь отчет.

Средства автоматизации

Генератор форм печати

Средства создания отчетов и этикеток имелись и в первой версии FoxPro, но в FoxPro 2.0 они были существенно переработаны. Во-первых, теперь вся информация, описывающая отчеты и этикетки, хранится в DBF файлах, структура которых описана в докумен-

тации. Это дает возможность программисту непосредственно воздействовать на форму получаемых документов. Кроме того, теперь можно вызвать, например, сразу несколько окон редактирования отчетов и копировать объекты из одной формы отчета в другую. Для временного хранения промежуточной информации при печати и других нужд в отчете могут быть определены переменные. Система предоставляет возможности легко манипулировать ими путем спецификации их начального значения и способа вычисления их значений при передвижении по базе в процессе печати.

Надо сказать, что FoxPro трактует объекты в формах печати не просто как необходимость печатать слева направо и сверху вниз, а в соответствии с порядком их логического следования. Это предоставляет пользователю ряд дополнительных возможностей, но одновременно требует повышенного внимания и понимания того, что будет происходить в том или ином случае. Мне часто приходится объяснять людям, отчаявшимся получить то, что они хотят, и считающим происходящее результатом ошибки в FoxPro, как сильно влияет на отчет порядок определения полей, групп и переменных. Один очень простой эксперимент даст Вам необходимую пищу для размышлений:

1. Создайте программу NXT с двумя командами:

```
SKIP
RETURN ""
```

2. Откройте любую базу с некоторым количеством записей.
3. Вызовите окно создания отчетов.
4. В области Detail создайте объект A типа Field, имеющий значение RECNO().
5. Правее создайте объект B типа Field, имеющий значение NXT().
6. Еще правее создайте объект C типа Field, имеющий значение RECNO() (можно выделить объект A, скопировать его в буфер, перевести курсор вправо и скопировать объект из буфера).

Выберите опцию Preview (предварительный просмотр) и посмотрите, что получилось. Естественно вот что:

```
1  2
3  4
...
```

А теперь поменяйте местами объекты A и C и опять посмотрите, что получилось:

```
2  1
4  3
...
```

Забавно, не правда ли?!

Еще одно новшество, связанное с печатью, обусловлено появлением в FoxPro драйвера принтера. Это, во-первых, дает пользователю возможность печатать разными шрифтами и начертаниями с учетом возможностей того или иного принтера, а во-вторых, самому создавать такой драйвер принтера, какой ему больше нравится, так как драйвер принтера в FoxPro 2.0 может быть обычной PRG программой, включающей в себя ряд процедур с предопределенными именами и параметрами. Это дает необычайно широкие возможности для организации вывода документов (например, печать на матричных принтерах в графическом режиме).

Генератор запросов

Великолепным средством построения запросов к базе с использованием SQL SELECT является появившаяся в FoxPro система RQBE. Система интуитивно понятна и позволяет использовать при построении запроса почти все возможности предложения SELECT. В то же время она является прекрасным средством обучения, так как позволяет тут же посмотреть, как будет сформировано предложение SELECT и каков будет результат запроса, в зависимости от выбора пользователем тех или иных управляющих опций. Результатом работы генератора является законченная программа, состоящая из предложения SELECT и, возможно, команд xBase.

Создание экранов

Нет больше программы FoxView, нет больше шаблонов с их ... языком! А есть превосходное средство, позволяющее в кратчайшее время создать экранную форму или даже целый набор связанных между собой окон для ввода и просмотра информации. Система великолепно организована и позволяет учесть практически все мыслимые желания пользователя и в то же время жестко структурирует и организует всю вводимую информацию. Система не позволяет себе никакой "отсебятины" и пользователь от начала до конца контролирует каждую строку получающейся потом программы.

Система фактически распадается на две подсистемы. Первая встроена в FoxPro и предназначена для создания самой формы и организации связанного с ней и вводимого пользователем программного кода. Результатом ее работы является DBF файл с документированной структурой, в котором хранится вся информация, связанная с формой. Другая часть представляет собой внешнюю программу, генерирующую исходный текст программы для формы и использующая для этого созданный ранее DBF файл. Так вот эта внешняя программа-генератор написана на языке FoxPro и поставляется в форме PRG файла. Вообще-то она и так генерирует прекрасно структурированный и откомментированный исходный текст. Но если кому-нибудь захочется ее подправить, ему больше уже не придется изучать какой-то там "шаблонный" язык (это, кстати, кое-что говорит о мощности языка FoxPro). Одним из вариантов такой правки может

быть, например, программа, генерирующая заготовку текста помощи по работе с данной формой (пусть и несколько формалистичную).

Создание меню

Система автоматизации создания меню предназначена для получения меню, подобного меню самого FoxPro, причем это касается как внешнего вида, так и функционирования. Дело в том, что FoxPro теперь позволяет сделать определенное пользователем меню системным, что, в свою очередь, позволяет целиком использовать поддерживаемый FoxPro интерфейс в проблемных программах (вплоть до создания программ, управляемых событиями). Последние подверсии FoxPro (начиная с 1992 года), кроме того, позволяют придать такому меню статус "по умолчанию". Процесс создания меню крайне прост. Пользователь указывает название опций, их тип, "горячие" клавиши, действия, связанные с выбором опции, и тому подобное. В ходе этого он имеет возможность в любой момент посмотреть, как будет выглядеть и функционировать созданное им меню. В остальном система получения меню аналогична системе создания экранных форм (включая и наличие программы-генератора).

Возможности команды READ

Команда READ расширена в трех направлениях. Первое определяется дополнительными возможностями команды @..GET и новой командой @..EDIT, тесно связанными с READ. Новые функции GET теперь позволяют включать в READ такие объекты, как командные кнопки, радиокнопки, индикаторы состояния, простые и раскрывающиеся поля выбора, невидимые области выбора и области редактирования. Таким образом программист получает возможность легко создавать программы с CUA интерфейсом, который используется в самом FoxPro или в Windows.

Вторая группа изменений связана с дополнительными опциями команды READ — ACTIVATE, COLOR, CYCLE, DEACTIVATE, LOCK, MODAL, NOLOCK, OBJECT, SHOW, VALID, WHEN, WITH. Достаточно подробно назначение каждой из опций изложено в документации. Отметим только, что теперь Вы можете размещать объекты READ в нескольких окнах, иметь одновременно доступ к системным окнам редактирования текстов или BROWSE или, наоборот, ограничивать доступные пользователю в процессе READ окна. Причем система создания экранных форм позволяет делать все это без особого напряжения.

И, наконец, команда READ теперь может употребляться вообще без объектов @..GET. Фирменное название такого READ — Foundation READ можно перевести как базисный READ. Используя такой READ и уже упоминавшуюся возможность придать определенному пользователем меню свойства системного меню, можно писать, следуя определенным правилам, программы, управляемые событиями (event-driven program interface).

Отличие программ с таким интерфейсом от более привычных (по крайней мере, в области работы с базами данных) программ с так называемым меню, управляемым интерфейсом, заключается в том, что пользователь не скован жесткой логикой, определяющей последовательность его действий. Это подобно сравнению встроенного в NC примитивного редактора, где для начала работы с новым текстом Вы должны сбросить текущий, с многооконным текстовым процессором. Это похоже на работу в среде DeskView или Windows, где каждая программа работает в своем независимом окне и пользователь переключается между ними просто путем активизации соответствующего окна или выбора опции меню, активизирующей это окно, без какой-либо связи с тем, какие еще окна присутствуют на экране, и, что важно, присутствуют ли они вообще. Разумеется, многозадачность при этом не поддерживается.

С учетом написанного о технологии Рашмора можно сказать, что мощность системы FoxPro 2.0 позволяет говорить об изменении самой парадигмы программирования в среде xBase.

Функции работы с массивами

В системе появилось 10 мощных и удобных функций, предназначенных для работы с массивами. Они позволяют копировать элементы массива в другой массив, добавлять и удалять элементы, производить сортировку элементов массива и осуществлять поиск в массиве. Кроме того, есть функции, позволяющие создать массив и заполнить его информацией из каталога файловой системы или информацией о структуре DBF файла.

Поддержка работы над проектом

При работе над проектом, включающем большое количество различных файлов: программ, баз, индексов, экранов, отчетов и тому подобных, которые, к тому же, могут находиться в разных директориях (например, библиотечные процедуры, кодификаторы и так далее), его становится очень трудно контролировать. Менеджер проектов, появившийся во второй версии, берет всю заботу по поддержанию целостности проекта на себя. Сам проект представляет собой опять-таки DBF файл документированной структуры, а менеджер проектов позволяет пользователю добавлять в него и удалять любые файлы, относящиеся к проекту. Впрочем, и это менеджер проектов может делать сам — пользователю нужно только указать головной файл программы и через меню или управляющую панель дать команду Rebuild Project (перестроить проект). В процессе перестройки менеджер сам отыскивает файлы по ссылкам на них, перегенерирует, в случае необходимости, тексты программ

экранов и меню и тому подобное. Для редактирования любого файла, включенного в проект, пользователю достаточно выбрать его из списка, и FoxPro автоматически подгрузит соответствующий типу файла редактор.

Помимо хранения информации о файлах проекта, менеджер позволяет также различным образом собрать все созданные программистом файлы (в том числе базы, индексы, формы отчетов, текстовые файлы и так далее) в один файл. Впрочем, пользователь может указать, какие файлы не должны включаться в этот файл. Сам FoxPro способен создать таким образом только файл типа APP, который может быть выполнен под управлением либо самого FoxPro, либо исполняющей системы, поставляемой отдельно в составе пакета FoxPro Distribution Kit. Этот дополнительный пакет дает лицензию и возможность для распространения разработанных в среде FoxPro 2.0 прикладных программ совместно либо с исполняющей системой, представляющей собой аналог систем RunTime для FoxBase или FoxPro 1.x, либо путем сборки всех файлов, составляющих проект в один EXE файл. Установка этого пакета дает менеджеру проектов возможность либо превратить APP файл в небольшой EXE файл (+8K), требующий для выполнения двух дополнительных файлов исполняющей системы (одного для расширенной версии), либо создать большой EXE файл (+ от 800K до 1200K), который уже ничего не требует для своего выполнения.

Интерфейс прикладных программ (API)

У FoxPro 2.0 появился низкоуровневый интерфейс, но, чтобы воспользоваться им, необходим пакет FoxPro Library Construction Kit. Пакет позволяет, используя компилятор Watcom C или ассемблер, создавать специальные библиотеки функций. Эти библиотеки представляют собой EXE файл специального формата и могут подсоединяться к FoxPro при помощи команды SET LIBRARY TO... [ADDITIVE]. Отключаются библиотеки либо при помощи той же команды, но без параметров — при этом отключаются все библиотеки, либо при помощи команды RELEASE LIBRARY... выборочно. Больше всего это напоминает DLL файлы Windows.

API FoxPro предоставляет программисту самый богатый набор интерфейсных функций среди систем этого класса. Он обеспечивает передачу в библиотечные функции параметров всех (строки, числа, даты, массивы) типов, как по значению, так и по ссылке, возврат результата любого типа в FoxPro, выделение и освобождение блоков памяти под управлением менеджера памяти FoxPro, манипуляцию окнами и меню, работу с базами данных, непосредственный доступ к memo полям, работу с файлами с использованием динамической буферизации FoxPro и многое другое.

Особо стоит отметить то, что API FoxPro дает программисту возможность перехватывать обработку событий, определенных в FoxPro, — пустые события, генерируемые 18.206... раз в секунду при отсутствии других событий; нажатие клавиши или кнопки "мыши"; активизация, деактивизация и уничтожение окон; выбор в меню и тому подобное. Можно также встроиться в цикл ожидания FoxPro (время, когда FoxPro ничего полезного не делает и ждет действий пользователя — нажатия клавиши, кнопки "мыши"). Все это позволяет создавать программы с весьма развитым интерфейсом и богатейшими возможностями (например, телекоммуникационный сеанс в фоновом режиме).

Такая открытость FoxPro подвигла многочисленные фирмы заняться коммерческими разработками различных библиотек. Уже к середине 1992 года предлагались десятки библиотек, специализированных в области деловой и иллюстративной графики, телекоммуникаций, доступа к базам Paradox, Oracle, Btrive, работе с серверами баз данных и так далее.

К сожалению, книжка, входящая в Library Construction Kit и посвященная описанию функций API, — самая тонкая из всей документации (если не считать книжку с исходными текстами примеров библиотек, существующих и в виде файлов). Недокументирован, например, ряд полей структуры, в которую FoxPro помещает информацию о событии. Написано, что нельзя вызывать на выполнение команды и функции FoxPro из функций, встроенных в очередь обработки событий, хотя фактически из них можно исполнить даже PRG программу и ничего плохого, на первый взгляд, не произойдет.

Две версии в одном пакете

FoxPro 2.0 выпускается в двух вариантах: однопользовательском и многопользовательском, предназначенном для работы в сети. Причем любой из этих вариантов содержит фактически две версии системы: стандартную, рассчитанную на любые IBM PC/XT совместимые компьютеры, и расширенную, предназначенную для компьютеров с процессором, совместимым с Intel 80386.

Стандартная версия требует для загрузки около 450K свободной памяти и может использовать до 32M дополнительной (expanded) памяти и блоки верхней памяти (UMB), расширенная версия работает в защищенном режиме 386 процессора, требует для работы минимум 1.5M памяти и может использовать всю имеющуюся в машине память.

В стандартной версии только основная память, блоки верхней памяти и первые 64K дополнительной памяти могут быть заняты пользовательскими объектами (программами, переменными, меню, окнами). Остальная дополнительная память используется только для буферизации операций ввода-вывода. В расширенной версии во всей доступной памяти могут храниться пользовательские объекты. По налагаемым ограничениям версии системы различаются следующим образом:

	Стандартная	Расширенная
Размер строки	64K	4Г
Количество переменных	3600	65000
Количество массивов	3600	65000
Количество элементов в массиве	3600	65000
Количество открытых файлов	99	без ограничений
Количество записей в файлах, участвующих в оптимизируемом запросе SELECT	500 000	без ограничений

Оптимизация работы

СУБД FoxPro 2.0 — подлинное произведение современного искусства в области программного обеспечения и написана она в расчете на "их" условия. Скорость работы FoxPro в значительной мере обусловлена исключительно эффективной технологией динамической буферизации операций ввода-вывода при работе с диском. При этом чем больше памяти доступно FoxPro, тем быстрее (в несколько раз) будет работать система. Не надейтесь, что на XT с 640K и 0K EMS FoxPro 2.0 обгонит FoxBase (если не учитывать технологию Рашмора). Главная рекомендация по оптимизации работы сводится к тому, чтобы любыми способами увеличить доступную FoxPro память. В стандартной версии, вне зависимости от процессора, постарайтесь увеличить свободную память, доступную DOS, и по возможности установите дополнительную (expanded) память 4-й версии.

Программы кэширования диска используйте, только если FoxPro вынужден часто открывать файлы в насыщенных директориях и размер оперативной памяти велик (>3M), но выделяйте под них не более 200-500K.

Вопрос о предпочтительности использования расширенной версии сложен, и его лучше решать экспериментальным путем. Обобщая, можно сказать, что, если память меньше 3M, надо использовать стандартную версию. В промежутке между 3M и 4M стандартная версия тем предпочтительнее, чем более интенсивна работа с базой и чем больше сама база (нужно только помнить об ограничении в 500000 записей для оптимизации SQL-SELECT запроса), а расширенная — чем более интенсивно используются интерфейсные возможности FoxPro. При дальнейшем увеличении памяти расширенная версия, как правило, предпочтительней. Все это становится понятным, если помнить, что у стандартной версии размер буфера ввода/вывода при размере памяти до 32M всегда больше, чем у расширенной, примерно на 1M, и что расширенная версия целиком грузится в память при наличии минимум 2.5M.

На 80386DX с памятью более 4M FoxPro 2.0 обгонит любую другую СУБД для ПК.

Проблемы русскоязычных пользователей и их решение

Целая группа проблем связана со специфической обработкой русской буквы "Н": упрощенно говоря, она плохо выводится на экран, выбрасывается при компиляции, не печатается в отчетах и так далее. С другой проблемой знакомы те, кто пишет программы с развитым интерфейсом, и сводится она к тому, что невозможно сослаться на окно, используя его заголовок (title), если он написан на русском. Это накладывает большое ограничение на интерфейс программы. Например:

```
BROWSE TITLE "Динамика цен на товар" NOWAIT
...
SHOW WINDOW Динамика REFRESH SAME
```

Последнюю команду FoxPro будет считать синтаксической ошибкой.

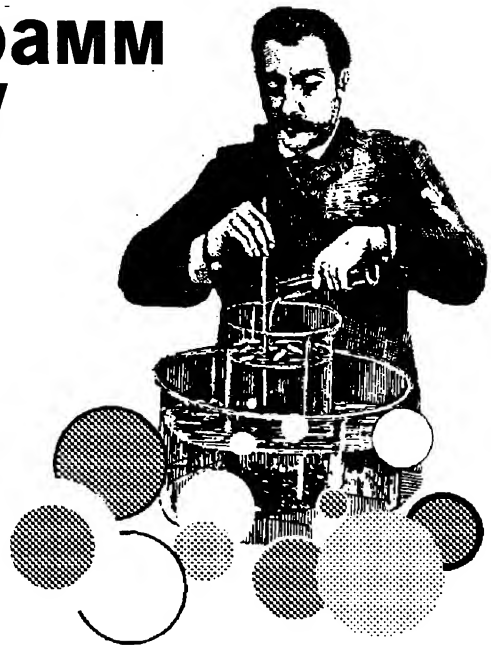
Вариант ПОЛНОГО решения всех "русскоязычных" проблем предлагает МП "НооСфера" (т. (095) 374-98-30). Созданный там русификатор FoxProR снимает не только описанные проблемы, но и вообще позволяет везде использовать русские буквы наравне с латинскими: в многочисленных строчных функциях FoxPro (большинство из которых, кстати, работает и с memo полями), в названиях окон, функций, процедур, переменных и полей базы данных (русские названия полей необходимы в программах, предоставляющих пользователю возможность самому создавать файлы баз данных, например, в системах статистической обработки произвольной информации или справочных системах). Все это касается не только различных вариантов и подверсий самого FoxPro 2.0, но и Distribution Kit и Library Construction Kit. Кроме того, "НооСфера" предоставляет своим пользователям "горячую телефонную линию" для консультаций по вопросам, связанным с FoxPro.

Что дальше

В конце января 1993 года появилась новая версия — FoxPro 2.5. Причем, в двух вариантах — для DOS и для Windows. Будут также варианты для Mac и UNIX. Интересно, что пользовательские программы смогут легко переноситься с одной платформы на другую, то есть APP, сделанный в DOS версии, сможет работать под Windows, и наоборот. При этом специфические для платформы возможности будут просто игнорироваться (например, связанные с шрифтами Windows в DOS версии). Новая версия будет выпускаться только в многопользовательском варианте. Судя по различным сообщениям, новая версия быстрее FoxPro 2.0 (на некоторых командах в 3 раза), имеет встроенный препроцессор и расширенные возможности оптимизации. С помощью предполагаемого к выпуску пакета FoxPro Connectivity Kit пользователи FoxPro смогут получать доступ к базам данных других систем. ■

Компиляция программ на языке dBASE IV

Ю. Сюганов



Кто никогда не держал его в руках, не поймет, о чем я говорю. От неожиданной мягкости пальцы проваливаются, разрывая то, что у обычной булки называлось бы корочкой. Это я про гамбургер. Впечатление от первого посещения ресторана McDonald's незабываемо. Здесь нравится все, кроме несколько назойливой музыки. Быстро, вкусно, удобно, после многих подорожаний — почти без очереди. Здорово, но это вовсе не значит, что у других ресторанов не остается никаких шансов.

С программным обеспечением примерно то же. Сколь замечательной ни была бы программа, ею не будут пользоваться все. Тому может быть множество причин: привычка, готовые файлы с данными в "старом" формате, необходимость переучиваться работе с новой программой и тому подобное. И пусть четвертая версия Paradox в 10 раз быстрее своей предшественницы и обогнала долго державший лидерство FOX. По общему количеству проданных экземпляров dBASE IV пока не догнал никто. То есть хоронить dBASE еще рано. А потому не может не радовать появление нового продукта для самой популярной в мире СУБД.

Речь идет о компиляторе для программ, написанных на dBASE IV.

Собственно, в качестве компилятора используется Clipper 5.0, а описываемый здесь пакет делает компиляцию возможной.

История вопроса

У языка dBASE существует несколько диалектов. Основа их общая. Стандарта языка пока нет. В конце 1992 года промелькнуло сообщение, что его собираются создать, но о сроках речь пока не идет. Наиболее известные (dBASE, FOX и Clipper) друг за другом поменяли хозяев. А вдруг новые и правда договорятся? Пока же — увы, увы...

Clipper был изначально создан для компиляции программ, написанных на языке dBASE. Программы, написанные на ранних версиях dBASE (по dBASE III Plus включительно), могут компилироваться с помощью Clipper без изменений. В Clipper, однако, было добавлено много новых свойств, а Clipper 5.0 — система разработки приложений, имеющая огромное число новшеств по сравнению с другими dBASE совместимыми системами создания приложений.

dBASE IV также имеет в составе своего языка много новых возможностей. В большинстве случаев эти новые команды и функции присутствуют и в Clipper, но имеют другие имена и синтаксис. И, к сожалению, приложение, написанное на dBASE IV, не может без большого количества изменений быть откомпилировано с использованием Clipper.

Новый продукт компании Computer Assosiates позволяет преодолеть разницу между этими двумя языками. Он состоит из набора утилит, переводящих команды, существующие только в dBASE IV, в команды, понятные Clipper. Полное название продукта — "Matt Whelan's Compiler Kit, dBASE IV Edition". Matt Whelan — это его автор. Compiler Kit можно перевести, как "средство для компиляции". dBASE IV Edition означает "версия для dBASE IV". Сокращенное название продукта, используемое в документации, — IV Kit. Далее я буду пользоваться именно им.

Интерпретаторы и компиляторы

Я не стану объяснять, чем отличается интерпретатор от компилятора и псевдокомпилятора. Если Вы этого еще не знаете, Вы, наверное, мастерски летаете на F29. Не подумайте, что язвлю. Несколько моих

приятелей успешно по нескольку раз долетались до генерала (тогда еще на F19), и уж не знаю, есть ли здесь какая-то связь, но стали вполне преуспевающими людьми. Так что у Вас еще многое впереди. (В электронной конференции после этого стояло бы сочетание трех символов — ":–)". Если Вы его впервые видите, эту конструкцию надо мысленно повернуть на 90 градусов по часовой стрелке.) А если серьезно, читайте врезку, где описана эта разница, а также преимущества и недостатки обоих подходов.

Комбинация dBASE IV, Clipper 5.0 и IV Kit позволяет объединить приемущества, свойственные обоим подходам. Вы можете создавать базы данных и программы в dBASE IV со всеми достоинствами интерпретирующей среды, а потом компилировать их, получая скорость и гибкость полностью компилированных систем.

Что такое IV Kit

В следующем далее тексте могут встретиться слова, смысл которых Вы не обязаны понимать сразу. Если такие действительно обнаружатся, не отчаивайтесь, попробуйте читать дальше. Очень может быть, что ниже им найдется какое-то объяснение. Итак,...

Работа компилятора Clipper состоит из двух фаз: фазы препроцессора и фазы компиляции. Препроцессор дает возможность компилятору понять синтаксис команд, не являющихся частью стандартного языка Clipper. Можно использовать специальные команды (они называются директивами препроцессора), чтобы определять новые команды и функции. Директивы препроцессора должны быть включены в исходный программный файл или в отдельный файл, который называют заголовочным, поскольку помещают его обычно в самом начале текста программы. Они показывают на фазе компиляции, как должны исполняться нестандартные команды и функции.

IV Kit использует препроцессор, чтобы сделать возможной компиляцию команд и функций, которые имеют отличающийся синтаксис вызова или вовсе отсутствуют в языке Clipper. Для переопределения этих команд IV Kit использует заголовочный файл.

Некоторые команды и функции dBASE IV просто переводятся в их эквиваленты на Clipper. Другие, для которых прямые эквиваленты отсутствуют, преобразуются в вызовы специальных откомпилированных функций IV Kit. Эти функции содержатся в библиотечном файле, который должен быть прикомпонован к программе вместе со стандартными библиотеками Clipper.

В файлах Ваших программ на dBASE IV должны быть инструкции для включения заголовочного файла IV Kit. В IV Kit есть утилита DPREP.EXE, которая вставит эти инструкции сама. DPREP, кроме того, делает еще некоторые изменения в исходном программном коде и создает файлы для автоматизации процессов компиляции и компоновки.

Итак, пакет IV Kit включает в себя:

- утилиту DPREP.EXE, которая обрабатывает исходный код на dBASE IV и создает файлы для автоматизации процессов компиляции и компоновки;
- заголовочные файлы, переводящие команды и функции dBASE IV либо в стандартные функции Clipper, либо в функции IV Kit;
- библиотечный файл, содержащий в компилированном виде функции, которые вызываются из заголовочного файла.

Полный процесс компиляции с использованием IV Kit

Далее следует описание полного процесса компиляции программ или приложений, написанных на dBASE IV, с использованием IV Kit и Clipper.

Перед компиляцией следует убедиться, что Ваши программы работают корректно в dBASE IV. Поначалу Вам, видимо, покажется удобнее исправлять ошибки в dBASE IV. Обретение некоторого опыта работы с языком Clipper приведет к тому, что... Впрочем, об этом позже.

Полный процесс компиляции изображен на рис.1.

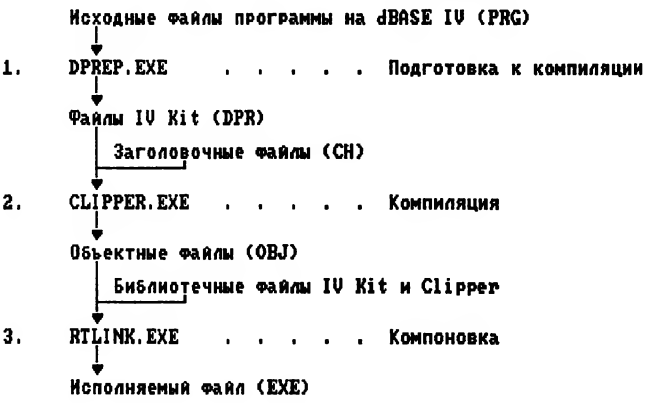


Рис. 1

Первый этап — запуск утилиты DPREP.EXE. Она скопирует исходные файлы dBASE IV (PRG, FRG, QBE, и так далее) в соответствующие файлы IV Kit (DPR), и сделает некоторые необходимые изменения в копиях. Оригиналы она не изменит, и они могут и дальше использоваться в dBASE IV, если это нужно.

Если Вы компилируете приложение, состоящее из нескольких файлов-источников, достаточно указать только основной (PRG) файл. Все другие файлы программ, которые вызываются из него, будут скомпилированы автоматически.

Второй этап — получение объектных файлов (OBJ) компилятором Clipper. Кроме DPR файлов, созданных на предыдущем шаге, понадобятся заголовочные (CH) файлы. DPREP сама поместит в начале DPR файлов инструкцию использовать заголовочные файлы во время компиляции.

И, наконец, третий этап: для получения исполняемого (EXE) файла запускается компоновщик (обычно

Что такое интерпретатор

Интерпретатор — интерактивная программа, которая, будучи однажды запущенной, позволяет вводить команды, которые и выполняет по мере поступления. Каждый раз введенная команда проверяется, транслируется в более эффективный машинный код и сразу же выполняется.

Поскольку даже не очень сложные операции требуют ввода больше чем одной команды, их можно записывать в текстовый файл. Получается программа. Когда Вы запускаете эту программу, интерпретатор считывает ее строку за строкой, воспринимая каждую, как если бы она была введена с клавиатуры. И Ваша программа не может быть исполнена, пока не загружена программа-интерпретатор.

Что такое компилятор

С компилятором процесс выглядит несколько иначе. Вы создаете программы в текстовом редакторе, как и в случае с интерпретатором, но после этого есть две раздельные фазы — трансляция и исполнение команд. Текстовый код (источник) проверяется компилятором на отсутствие в нем ошибок и если их нет, транслируется в более эффективную форму (объектный код). После этого объектный код компонуется с другими уже скомпилированными программами и библиотеками функций, чтобы получить исполняемый файл. Эту финальную версию программы можно запустить на выполнение уже из операционной системы компьютера, набрав ее имя в командной строке DOS. Никакая другая программа (интерпретатор) для этого уже не нужна.

Техника, используемая в dBASE IV, — нечто среднее между этими двумя подходами. Когда программа запускается в первый раз, происходит псевдокомпиляция — перевод ее в другую форму. При следующем запуске она не повторяется, если в исходный файл не вносились никакие изменения. Однако dBASE IV (или RunTime модуль, способный читать и выполнять подвергшиеся такой обработке программы) должен присутствовать в памяти компьютера каждый раз, когда Вы запускаете свою программу.

Преимущества интерпретатора

Оба подхода имеют преимущества и недостатки. Преимущества интерпретатора состоят в том, что

- код может быть проверен до включения в программу вводом команд по одной в командной строке; когда используется компилятор, весь код должен быть сначала оттранслирован и скомпонован с библиотеками, прежде чем можно будет посмотреть, что же получилось;
- файлы данных могут легко создаваться интерактивным вводом соответствующих команд;
- программу можно отлаживать и быстро смотреть результат, запуская ее на выполнение.

Интерпретаторы и компиляторы

Преимущества компилятора

Преимущества компилятора по сравнению с интерпретатором состоят в том, что

- программы выполняются значительно быстрее. Интерпретатор вынужден транслировать каждую команду столько раз, сколько Вы запускаете свою программу. Компилированная программа хранится в транслированной форме, и транслировать ее снова не надо;
- компилированная программа не требует для своего выполнения другого программного обеспечения. Если Вы продаете созданное Вами приложение, установить на чужой компьютер надо только Ваш исполняемый файл. Интерпретируемая программа не сможет работать без интерпретатора;
- компилированная программа не поддается легкому изменению, поскольку Вы не поставляете исходные тексты. В интерпретируемой программе изменения может делать кто захочет;
- процесс компиляции обеспечивает большую совместимость с другими программными системами. Многие языки программирования используют одинаковый процесс компиляции-компоновки, и часто возможно объединение программ, написанных на разных языках. Кроме того, компилированные программы взаимодействуют напрямую с операционной системой, поэтому проще расширять возможности языка, обеспечивая непосредственную связь с программным окружением. Интерпретаторы, наоборот, обычно закрытые системы. Из них чаще всего трудно или вообще невозможно взаимодействовать напрямую с другими программами.

dBASE IV, в противоположность более ранним версиям dBASE, обеспечивает некоторые из этих преимуществ своим псевдокомпилятором. Программы выполняются быстрее (хотя и не так быстро, как скомпилированные Clipper) и не могут быть изменены посторонними лицами.

Комбинация dBASE IV, Clipper 5.0 и IV Kit обеспечивает Вам все преимущества обоих подходов. ■

компоновщик Clipper, RTLINK.EXE, однако Вы можете использовать другой компоновщик). Компоновщику на входе нужны объектные (OBJ) файлы, и библиотечные файлы Clipper и IV Kit.

Процесс может показатсья сложным, но DPREP упрощает его, создавая два файла, позволяющих пройти этапы 2 и 3 (компиляцию и компоновку) автоматически. Во-первых, он создает RMK файл. Этот файл используется утилитой Clipper RMAKE, которая управляет компилятором и компоновщиком. Во-вторых, создается LNK файл, передающий правильные имена файлов на вход компоновки. Эти файлы называются по умолчанию IVKIT.RMK и IVKIT.LNK.

Кроме того, IV Kit снабжен пакетным файлом COMPILE.BAT, который вызывает утилиты DPREP (1-й этап) и RMAKE (2-й и 3-й этапы). Таким образом, весь процесс получения исполняемого файла Вашего приложения сводится к командной строке, где команде COMPILE передается в качестве параметра имя основного файла Вашего приложения, то есть если он называется MyApp.prg, в командной строке достаточно набрать

```
COMPILE MyApp <Enter>
```

Ограничения и отличия в работе

Разработчиком IV Kit были предприняты немалые усилия, чтобы сделать его насколько возможно совместимым с языком dBASE IV, и он в этом преуспел. Однако между языками dBASE IV и Clipper существует ряд отличий, которые заставят утилиту DPREP выдать предупреждающие сообщения или приведут к тому, что откомпилированная с помощью IV Kit программа будет выполняться в некоторых деталях иначе, чем в режиме интерпретации dBASE IV. Это означает, что Вам придется внести изменения в исходный текст программы перед компиляцией. Никогда не возникнет необходимости переписать заново большой фрагмент программы; скорее всего, придется вставить, удалить или изменить несколько команд. На эти места будут даны подробные ссылки: имя файла, строка, что DPREP в ней не понравилось, и зачастую совет, на что ее следует заменить.

Я не вижу смысла приводить здесь полный список команд, которые не рекомендуется использовать с IV Kit: он есть в документации, но несколько слов по этому поводу все же скажу.

Clipper, в отличие от dBASE IV, — не интерпретирующая среда, а компилятор, запускаемый из командной строки DOS. В нем, естественно, отсутствуют команды, имеющие смысл при работе в режиме диалога и управляющие параметрами именно диалоговой среды. Например, команда SET SAFETY ON/OFF, которая включает и выключает выдачу сообщения о попытке перезаписать уже существующий файл. Само собой разумеется, что вид этого сообщения стандартный, то есть такой, каким он показался оптимальным создателям dBASE IV. И, конечно же, на английском языке. В Clipper всем экранным выводом управляет программист, и все сообщения появляются там и тогда, где и когда он считает это нужным.

Или, например, команда SET STATUS ON/OFF, управляющая строкой статуса, тоже отсутствует в Clipper, поскольку нет такого понятия, как статусстрока. То есть если автор программы считает нужным отображать текущее состояние системы (номер текущей записи, имя открытого DBF файла и тому подобное), он вправе это делать теми средствами, которые больше отвечают его представлениям о прекрасном. В dBASE IV Вам дают возможность только совсем отключить эту строку статуса. (Чувствую гнев поклонников dBASE. Немножко преувеличил. Да и ладно...)

Далее, в Clipper отстутствует встроенная автоматическая поддержка работы в сети. Из-за этого о необходимости ограничить право других пользователей вносить изменения в файл, когда Вы, например, суммируете содержимое какого-то поля, должен думать программист, а не сам Clipper. Все необходимые для этого средства в языке есть. Или, например, систему ограничения доступа к файлам по паролю в Clipper программисту предлагается делать самостоятельно, в то время как в dBASE IV для этого есть встроенные команды PROTECT и LOGOUT. Это может нравиться или совсем наоборот. Можно считать, что в Clipper нет важного для Вас свойства. Или, напротив, относить к его достоинствам то, что он не навязывает Вам своего решения, то, что позволяет написать это наиболее Вам подходящим способом. В любом случае, когда Вы компилируете написанную на dBASE IV программу, об этих отличиях двух языков следует помнить.

Еще одно замечание: IV Kit не поддерживает полноэкранные команды создания и редактирования файлов, экранов ввода, отчетов, этикеток, запросов по образцу, автоматической генерации программ, то есть всего того, что не поддерживает и RunTime модуль самого dBASE IV.

Наконец, заслуживающая внимания особенность – разные области видимости процедур в dBASE IV и Clipper.

В dBASE IV процедура (функция) считается принадлежащей тому и только тому файлу, в котором определена. Это дает возможность в разных файлах иметь одинаково называющиеся процедуры, которые делают похожие (иначе нет смысла их одинаково называть), но отличающиеся в деталях вещи. Это, например, дает возможность определить одного вида меню в одном файле и вызывать его из разных, не теряя уверенности, что одноименный пункт меню, например, "Фильтр", будет выполнять фильтрацию той базы, с файлом которой работает соответствующий фрагмент программы.

В Clipper же процедура может вызываться из любого места и определена может быть в любом файле, а из двух одноименных второе определение просто бу-

дет проигнорировано. Это, кстати, дает возможность переписывать стандартные функции, например, GET систему, в соответствии со своими потребностями. (При компоновке "свой" файл надо будет указать первым.)

Тех, кому нравится идея хранения индексной информации для разных ключевых выражений в одном файле, видимо, интересует вопрос о поддержке "многоиндексных файлов" (Multiple Index Files — MDX). Поддерживаются! В Clipper (специально для возможности использования разных форматов файлов данных и индексов) применяется механизм замещаемых драйверов баз данных. Применительно к рассматриваемой ситуации это означает, что для поддержки MDX файлов надо просто прикомпоновать входящий в поставку файл DbfMdx.Lib.

Само собой разумеется, поддерживаются и стандартные индексные файлы Clipper NTX.

Состав поставки. Документация

Пакет IV Kit поставляется в двух разных вариантах: с компилятором Clipper (для тех, кто не является его зарегистрированным пользователем) и без него. Первое удовольствие, разумеется, дороже.

Полный комплект включает в себя 10 дискет по 360 Кбайтов и книгу с описанием. На дискетах — утилита DPREP, заголовочные и библиотечные файлы IV Kit, полный комплект Clipper, электронный справочник по системе в формате Norton Guide и пример приложения на dBASE IV, которое предлагается использовать для обучения. Есть еще и файл READ.ME, содержащий самые последние сведения, не вошедшие в состав документации.

В книге 2 раздела, приложения, список терминов с определениями и индекс (ключевые слова по алфавиту с номерами глав и страниц, где они употреблены).

В первом разделе подробно описаны:
- установка пакета на Ваш компьютер;
- основные этапы компиляции, приведенные в этой статье;
- шаг за шагом — компиляция первого приложения;
- использование справочника Norton Guide.

Второй раздел — очень подробный справочник по утилите DPREP.EXE, утилитам Clipper.EXE, RTLINK.EXE и RMAKE.EXE и, наконец, описание различий языков dBASE IV и Clipper и изменений, которые DPREP вносит в копии исходных текстов программ.

В приложении:
- список сообщений об ошибках во время выполнения и причинах, их порождающих;
- список зарезервированных слов, которые не могут быть использованы в качестве имен переменных и пользовательских функций. (Надо заметить, достаточно обширный, на 5 страницах, здесь все функции Clipper и IV Kit.)

Поверхностные впечатления

Я уже упомянул, что в состав поставки входит демонстрационный пример. Было бы наивно думать, что его компиляция вызовет какие-то проблемы. Не вызвала.

Скорость работы программ (внутри dBASE IV и в виде EXE файла) отличалась так, что это было заметно на глаз. Размер исполняемого файла получился 625 Кбайтов. Он без проблем выполнялся поверх Norton Commander, который, в свою очередь, сообщал, что доступно 537 Кбайтов свободной памяти.

Если представить, что эта программа (я говорю о демонстрации) написана Вами и Вы собираетесь ее продавать (оставим в стороне вопрос "зачем она покупателю?") с модулем RunTime.exe dBASE IV, с которым только Вы и имели бы право это делать, Ваш пакет занимал бы объем чуть больше одного мегабайта. Про быстродействие я уже сказал.

Чем все это кончится

Естественно, иногда, хотя бы из любопытства, Вы станете заглядывать в код на Clipper, полученный из Вашего исходного текста на dBASE IV. Возможно, Вы станете просматривать документацию Clipper, и не исключено, что однажды Вас посетит не такая уж страшная мысль: а не перейти ли на Clipper совсем?

Если перспектива зарабатывать на жизнь созданием баз данных Вам не кажется уж очень забавной (нормальные брокеры, не будем спорить, получают больше), то в итоге Вам обязательно понадобится та гибкость, которой dBASE, конечно, не дает. (Под гибкостью здесь я подразумеваю степень зависимости всего, что появляется на экране, только от нажатых перед этим клавиш и Ваших замыслов. Только от них.)

В этом случае в качестве моста для плавного перехода из любителей в профессионалы IV Kit окажет Вам неоценимую услугу. ■

Журнал "Монитор" можно приобрести в городах России, Украины, Беларуси, Узбекистана, обратившись к нашим официальным дилерам.

1. **Самара**, *МП "Сети", директор Дмитрий Грецов*
   Тел. (8462) 354-932
2. **Волгоград**, *арендное предприятие "Техническая книга"*
   Тел. (8442) 363-597
3. **Чебоксары**, *фирма "Демос"*
   Тел. (8352) 208-395, 426-492, факс 209-437
4. **Киев**, *МП "Диас"*
   Тел./факс (044) 220-77-15, тел. 518-80-24
5. **Днепропетровск**, *Попов А.Е.*
   Тел. (0562) 428-593
6. **Минск**, *НПП "Триумф"*
   Тел. (0172) 661-609, 661-680, факс 666-606
7. **Ташкент**, *Центр молодежных исследований*
   Тел. (3712) 688-847
8. **Тамбов**, *АО "От Елены" ЛПС "Shift"*
   Тел. (0752) 22-44-63

SYMANTEC

ZORTECH

Мощный компилятор языка C++ и набор инструментальных средств для создания превосходных прикладных программ для платформ Windows™ , DOS и OS/2 v 1.x

3.1

Поддержка Windows 3.1

Библиотеки Windows SDK

Полная поддержка языковых стандартов ANSI C, AT&T C++ 3.0

Прекомпилированные заголовки

В комплекте – популярный отладчик Multiscope 2.0

Whitewater Resource ToolKit

Исходные тексты библиотек

Генерация 32–разрядных кодов!!!

Дистрибьютеры Symantec:
Диалог–МИФИ – 095–320–3466
Merisel Russia – 095–276–9098
Перспективные технологии– 095–256–6271
Трио Плюс – 095–971–1204

Монитор — 1993

Подписной купон

МОНИТОР

Адрес _____

Ф.И.О. или название организации _____

_____

Стоимость подписки на журнал "Монитор" на 1993 год — 1700 рублей. Перечислите сумму оплаты на счет НТО "Софт-Москва": р/с 467002 в КБ "Банк развития XXI век", к/с 161816 в РКЦ ГУ ЦБ РФ по г. Москве, МФО 201791.
Вышлите подписной купон и квитанцию об оплате по адресу: 117419 Москва, а/я 765.

Монитор-Аспект — 2.93

Подписной купон

МОНИТОР Аспект

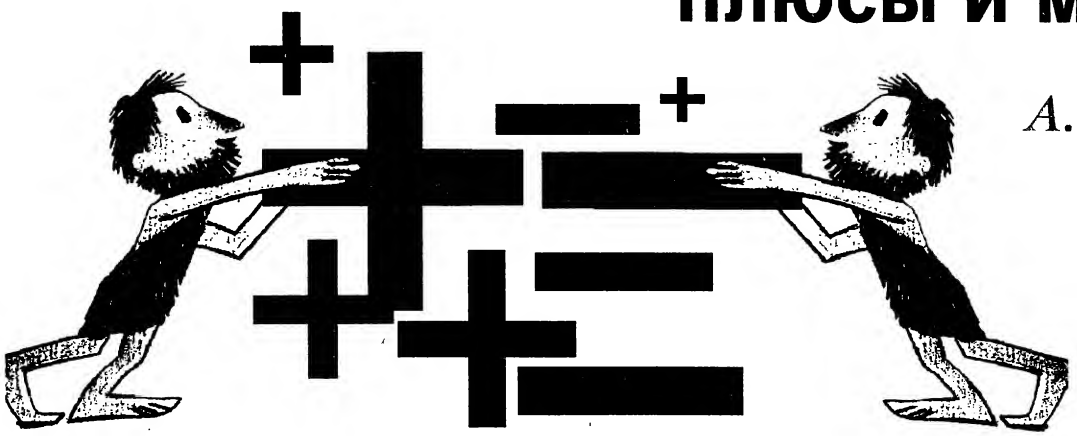
Адрес _____

Ф.И.О. или название организации _____

_____

Стоимость подписки на журнал "Монитор-Аспект" № 2.93 — 250 рублей. Перечислите сумму оплаты на счет НТО "Софт-Москва": р/с 467002 в КБ "Банк развития XXI век", к/с 161816 в РКЦ ГУ ЦБ РФ по г. Москве, МФО 201791.
Вышлите подписной купон и квитанцию об оплате по адресу: 117419 Москва, а/я 765.

Расширенная сетевая модель данных MDBS III: плюсы и минусы



А. Карев

Все наиболее известные системы управления базами данных (СУБД) для персональных компьютеров (dBASE IV, Clipper, Paradox, FoxPro) основаны на реляционной модели данных. Они предлагают пользователю рассматривать базу как совокупность двумерных однородных таблиц (файлов) и предоставляют ему средства для их просмотра, разрезания, склеивания и так далее. Кроме реляционных СУБД, существуют системы, использующие совершенно иную, сетевую модель данных, ведущую начало от проекта DBTG CODASYL, впервые опубликованного еще в 1969 г. Примерами таких СУБД могут служить пакеты MDBS III (фирма Micro Data Base Systems, Inc.) и db_VISTA III (фирма Raima Corp.). Причем если db_VISTA III использует классическую сетевую модель данных, то MDBS III основывается на расширенной сетевой модели. На примере пакета MDBS III рассмотрим структуру и возможности сетевых СУБД.

Сетевая модель данных

Итак, основным отличием сетевых СУБД от реляционных является логическая структура модели данных. При построении логической структуры базы данных (БД) очень важно для прикладного программиста иметь некоторые средства для определения отношений (связей) между типами записей. В реляционных моделях это достигается за счет избыточности: один или более элементов данных в одном типе записи повторяются в другом типе записи. Экземпляры одного типа записи, которые связаны с данным экземпляром другого типа записи, могут быть затем найдены посредством поиска на совпадение значений между повторяющимися элементами данных. В сетевых моделях данных взаимосвязь между записями устанавливается непосредственно введением их в набор и реализуется с помощью индексированных динамических массивов указателей. Термин "set" (набор, множество), как он используется в управлении базами данных, не имеет связи с математическим понятием множества, а определят характер взаимосвязи между записями БД.

Основу структуры модели данных в MDBS III составляют следующие понятия:

1. Поле данных — элементарная составляющая БД. У каждого поля данных есть имя, тип, размер его значений.

2. Тип записи — группа с наименованием, состоящая из полей данных.

3. Набор — связь между двумя типами записи (каждому набору присваивается имя).

Один тип записи, входящий в набор, называется владельцем набора, а другой тип записи называется членом набора. В MDBS допускается 4 вида наборов:

1:1 — набор для представления связи "один к одному" (владелец-член);

1:N — набор для представления связи "один ко многим" (владелец-члены);

N:1 — набор для представления связи "многие к одному" (владельцы-член);

N:M — набор для представления связи "многие ко многим" (владельцы-члены).

В структурах MDBS каждый тип записи может неоднократно входить в различные наборы в качестве их членов или владельцев. На рис. 1 показан пример ло-

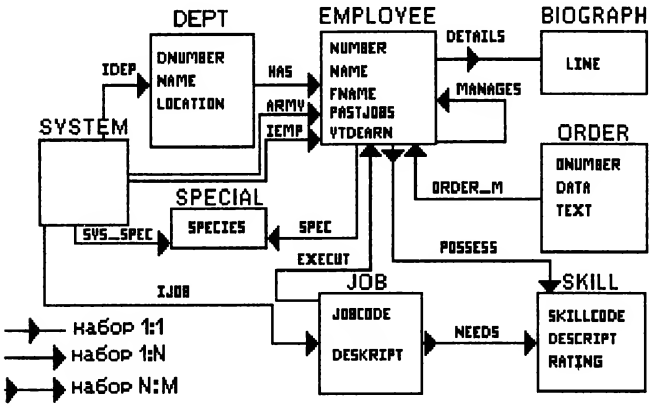


Рис. 1. Пример логической структуры данных

гической структуры БД, содержащей информацию обо всех отделах организации. Для каждого отдела мы хотим получить номер отдела, название и размещение. Эта информация содержится в записи DEPT с соответствующими полями данных DNUMBER, NAME, LOCATION. На рисунке все записи обозначены прямоугольником, над которым записано имя записи, а внутри имена полей данных. Набор обозначен линией между типом записи владельца и типом записи члена. Имя набора помещено вдоль линии. Стрелка на линии всегда направлена от владельца к члену, а ее положение определяет вид набора. Кроме данных об отделе, БД в нашем примере хранит информацию о служащих (запись EMPLOYEE), их биографиях (BIOGRAPH), выполняемых ими заданиях (JOB), навыках (SKILL), специальностях (SPECIAL) и приказах, относящихся к ним (ORDER). Рассмотрим на этом примере использование всех четырех видов наборов.

Набор 1:N — это обычный набор CODASYL. Определяя набор 1:N, прикладной разработчик указывает, что один экземпляр типа записи владельца может быть ассоциирован с N (N > 0) экземплярами типа записи члена. Однако экземпляр типа записи члена ассоциируется не более чем с одним экземпляром типа записи владельца. На рис. 1 HAS является примером набора 1:N. Один отдел может иметь много служащих, но один служащий связан не более чем с одним отделом. EXECUT — другой набор 1:N, тип задания может выполняться многими служащими, но один служащий выполняет только одно задание в какой-то момент времени.

Набор N:M между двумя типами записи означает, что экземпляр типа записи владельца может ассоциироваться с M (M > 0) экземплярами типа записи члена. Кроме того, экземпляр типа записи члена может ассоциироваться с N (N > 0) экземплярами типа записи владельца. Поэтому набор N:M используется для непосредственного представления связи "многие ко многим" между типами записей. На рис. 1 NEEDS представляет набор N:M. Для выполнения задания необходимы различные навыки, а один навык может понадобиться для многих заданий. POSSESS является другим набором N:M: служащий может иметь много навыков, а одинаковым навыком могут обладать многие служащие.

Набор 1:1 между двумя типами записи означает, что экземпляр типа записи владельца ассоциируется не больше чем с одним экземпляром типа записи члена, и экземпляр типа записи члена ассоциируется не больше, чем с одним экземпляром типа записи владельца. На рис. 1 DETAILS является примером набора 1:1. Служащий имеет только одну биографию, а биография ассоциируется только с одним служащим. LINE является строковым элементом данных (то есть с переменной длиной), который мог бы иметь очень длинный экземпляр для одних служащих и очень короткий для других. LINE может быть включен в тип записи EMPLOYEE. Однако если эти данные используются намного реже, чем другие данные о служащем, было бы разумно поместить LINE в отдельный тип записи. Это означает, что длинные биографические данные не будут вноситься в основную память каждый раз, когда выбирается запись EMPLOYEE. Также разрешается назначать биографическую информацию той области, которая, как правило, не является оперативно доступной и поэтому находится на дисководе с медленной выборкой.

Набор N:1 является точной обратной копией набора 1:N. Для набора N:1 экземпляр члена может иметь N (N > 0) экземпляров владельца. Однако экземпляр владельца ассоциируется не больше чем с одним экземпляром члена, например, набор SPEC.

Экземпляр набора состоит из экземпляра типа записи владельца набора и всех ассоциированных экземпляров записи члена (например, отдел и все служащие этого отдела; другой пример — задание и все навыки, которые нужны для его выполнения). Есть возможность внутри любого набора задать логическую упорядоченность его членов и владельцев, определяющую последовательность выборки записи.

До сих пор набор использовался для представления связи между двумя различными типами записи, определенными прикладным разработчиком. Существует четыре других способа для использования наборов: наборы, относящиеся к системе, рекурсивные наборы (множества), наборы с множеством типов записей члена, а также наборы с множеством типов записей владельца. В трех последних случаях допускается использование всех четырех типов наборов. В случае использования системных наборов разрешаются только наборы типа 1:1 и 1:N. Также существует специальный набор 1:N с названием $SYSSET, который автоматически определяется во время инициализации базы данных.

Системный набор

Во время инициализации базы данных автоматически создается специальный тип записи, который называется SYSTEM. Этот специальный тип записи имеет один экземпляр записи в базе данных. Он не

содержит данных. Набор, принадлежащий системе,— это набор, который имеет SYSTEM в качестве владельца и какой-то другой тип записи в качестве его члена. На рис. 1 показано пять наборов, принадлежащих системе: SYS_SPEC, IEMP, IDEP, IJOB, ARMY. Тип записи может быть членом ни одного, одного или большего числа наборов, принадлежащих системе. Объявление сортируемого системного набора заставляет MDBS автоматически формировать и поддерживать индекс в экземплярах типа записи члена. Для повышения эффективности выполнения этот индекс берется как многоуровневый сбалансированный. Индекс поддерживается в соответствии с логическим порядком члена, объявленного для системного набора. Предположим, что IEMP сортируется по FNAME и NAME. Тогда сортируемый список (по фамилии и имени) служащих может быть быстро воспроизведен с использованием IEMP. Это также позволит быстро найти экземпляр конкретного служащего на основе значения ключа сортировки. Можно объявить много наборов системы для одного и того же типа записи члена.

Рекурсивный набор

Набор является рекурсивным, если он имеет один и тот же тип записи и в качестве владельца, и в качестве члена. Он является средством для связывания экземпляров типа записи с другими экземплярами этого же типа. Например, служащий может управлять многими другими служащими, которые в свою очередь управляют другими служащими, и так далее. Таким образом, существует связь "один ко многим" среди служащих. Эта рекурсивная ситуация легко обрабатывается рекурсивным набором MANAGES, который изображен на рис. 1. Как и любой другой набор 1:N, MANAGES может иметь сортируемый порядок членов. Необходимость в рекурсивных наборах 1:N и N:M возникает в различных прикладных сферах (например, таблицы расчетов в бухгалтерии).

Разветвленные наборы с множеством членов

Как в случае с сетями CODASYL, расширенные сети MDBS допускают использование наборов 1:N, которые имеют множество типов записей членов. В случае с MDBS множество типов записей членов также разрешается для наборов 1:1, N:1 и N:M. На рис. 2а приводится пример разветвленного набора 1:N с множеством типов записей членов. Этот набор называется CONTAINS. Так как представлен набор 1:N, то экземпляр записи DEPT может иметь много экземпляров записей членов, некоторые из которых типа HRLYEMP и некоторые типа SALEMP. Отметим, что два типа записи члена имеют как общие элементы данных, так и специфичные. Все обычные порядки наборов допускаются в наборе множественных членов. Например,CONTAIN может сортироваться (в возрастающем порядке) по FNAME и NAME. К записям о служащих данного отдела можно затем обратиться в алфавитном порядке, независимо от вида получаемой зарплаты (сдельная или повременная). Кроме того, имя типа записи может включаться как часть в ключ сортировки. Если CONTAINS сортируется сначала по имени типа записи (в порядке возрастания), а потом по FNAME и NAME, затем по данному отделу, то тогда можно обратиться к алфавитному списку сотрудников, получающих сдельную зарплату, за которым следует алфавитный список служащих с повременной оплатой.

Разветвленные наборы множественных владельцев

В MDBS допускается более чем один тип записи в качестве владельца любого набора. На рис. 2б представлен пример разветвленного набора N:M с типами записи множественных владельцев.

Набор $SYSSET

Это специальный набор 1:N, который автоматически определяется при инициализации базы данных. Его владельцем является SYSTEM. MDBS автоматически и динамически устанавливает (и переустанавливает) тип (типы) записей членов во время выполнения прикладной программы.

Примерное описание логической структуры БД (рис. 1) в терминах языка описания данных MDBS III представлено в листинге 1. В соответствии с таким описанием анализатор СУБД инициализирует базу данных.

Языки доступа

Фирма-разработчик СУБД MDBS III с самого начала ориентировалась на определенный круг пользователей, а именно прикладных программистов-профессионалов. Отсюда и стратегия создания прикладных БД на языках, наиболее широко используемых в программировании: Кобол, Паскаль, Бейсик, Фортран, Турбо Си, Си++ и ассемблер. Разработчик имеет воз-

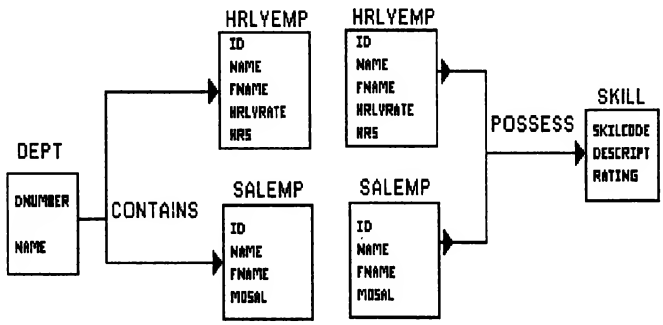


Рис. 2а. Разветвленный набор с множеством членов

Рис. 2б. Разветвленный набор с множеством владельцев

можность не обучаться "новому" языку, а использовать все свои навыки и наработки, программируя на своем любимом языке. При этом выполнение операций над БД сводится к вызову из прикладной программы отдельных функций системы манипулирования данными MDBS (Data Manipulation Language — язык манипулирования данными). Это одно из достоинств системы MDBS III.

Язык манипулирования данными

В MDBS III язык манипулирования данными состоит из группы команд, каждая из которых выполняет некоторое определенное задание по манипулированию данными. Все команды разбиты на 14 классов по их целевому назначению: поиск записей, выборка данных из записей, модификация данных в записях, присвоение индикаторов текущего состояния, создание записей, соединение записей, разъединение записей, удаление записей, команды доступа к логической схеме, обслуживающие программу, булевские операции, специальные операторы, многопользовательская блокировка, операторы восстановления. В рамках одной статьи невозможно описать все команды, но пример вызова из прикладной программы (язык доступа C++) некоторых наиболее часто используемых команд представлен в листинге 2.

Преимущества и недостатки

При структуризации данных в сложных задачах, требующих хранения большого количества разнотипной многосвязной информации, использование сетевой модели данных приносит пользователю ряд преимуществ по сравнению с реляционными СУБД. Во-первых, использование двумерных таблиц в реальной прикладной разработке ведет к увеличению требуемой памяти для представления отношений между записями. Например, для представления отношения между записями типа DEPT и EMPLOYEE (рис. 1) в реляционной модели необходимо одно и то же поле (например, DNUMBER) поддерживать в обеих записях в качестве ключа связи. При решении реальных задач обычно используется много типов записей, между которыми устанавливается множество разнообразных связей, при этом каждая из них требует такого дублирования. В результате при большом количестве экземпляров записей в сложной СУБД расход памяти весьма заметен. Сетевая модель данных лишена такого недостатка, так как связь устанавливается путем введения экземпляра записи в набор (например, HAS) в качестве владельца или члена набора. При этом использование динамических массивов указателей для связывания записей позволяет избежать каких-либо затрат на отсутствующую связь. В реляционных же моделях независимо от того, связаны ли два конкретных экземпляра записи или нет, сама возможность связывания требует дублирования полей.

Во-вторых, в реляционных системах при выборке связанной информации из БД требуется дополнительное время на обработку запроса. Например, по запросу типа "выбрать все приказы на Иванова" необходимо просмотреть всю таблицу записей типа EMPLOYEE, найти запись с полем типа FNAME и значением "Иванов", получить значение поля NUMBER , а затем, просматривая всю таблицу ORDER, искать записи, содержащие в поле NUMBER то же самое значение. Таким образом производятся два просмотра таблиц, две операции извлечения значения ключевого поля и сравнения его значения. В сетевых моделях такой запрос реализуется следующим образом: дается одна команда, производящая поиск члена набора IEMP на основе значения поля типа FNAME, равного "Иванов" (в MDBS это команда FMI). В случае нахождения требуемой записи дается команда типа "установить член набора ORDER_M, равный члену набора IEMP" (err=SMM("ORDER_M,IEMP") — пример вызова команды из прикладной программы, написанной на C++), тем самым все владельцы в наборе ORDER_M, связанные с установленным членом, становятся доступны, то есть выбираются все приказы, связанные с конкретным экземпляром записи EMPLOYEE. При необходимости осуществлять выборку по нескольким связям и в определенной логической последовательности реляционная модель поиска еще более усложняется.

В третьих, на реляционных моделях данных невозможно осуществить множественное связывание записей в случае неизвестного количества возможных связей одной записи. Этот случай возникает при моделировании различных справочников. Например, в записи типа SPECIAL (рис. 1) хранится справочник по всем возможным видам специальностей, каждая запись EMPLOYEE может быть связана с любым экземпляром записи SPECIAL, а каждый экземпляр записи SPECIAL может быть соединен с любым количеством экземпляров записи EMPLOYEE. Поскольку заранее неизвестно количество карточек и тем более неизвестно, сколько служащих имеют одну специальность, то невозможно определить количество ключевых полей, необходимых для связывания этих записей. Можно, конечно, определить максимально возможное число, но тогда такая БД будет слишком неэффективно использовать память, а в большинстве случаев это практически невозможно. В сетевых моделях таких проблем просто не возникает. Например, в примере на рис. 1 множественное связывание осуществляет набор SPEC с типом N:1. Кроме того, благодаря объявлению системного набора SYS_SPEC очень просто реализуются функции, необходимые для работы со справочником (листинг 2).

В четвертых, в сетевых моделях очень удобно использовать системные наборы для представления бинарных отношений. На рис. 1 набор ARMY представляет такую связь, причем, если служащий является военнообязанным, то его EMPLOYEE включается в набор, если не является военнообязанным, то запись EMPLOYEE не включается в набор ARMY. Такой

подход позволяет уменьшить количество типов записей в БД, а следовательно, уменьшить занимаемую память и просто проводить выборки. Например, необходимо выбрать всех военнообязанных служащих, это достигается выборкой всех членов набора ARMY (вызов в цикле функции FNM()). Такие возможности в реляционной модели отсутствуют.

В пятых, наличие явного описания связей между записями упрощает разработчику представление полной структуры сложной БД и облегчает ориентацию в ней. Между тем, в реляционных структурах не всегда легко выделить связи между типами данных, особенно когда два типа записи косвенно связаны посредством нескольких других типов записей. Эта проблема касается реальных логических структур, которые обычно вовлекают много типов записей (не 2 или 3). Недостатком расширенной сетевой модели данных является более высокая сложность освоения по сравнению с реляционной моделью данных. Этот недостаток легко преодолим, так как MDBS III поддерживает описание данных в соответствии с реляционной моделью. Поэтому разработчик СУБД, не овладевший всеми преимуществами сетевой модели, может остановиться на более простом варианте. Важно помнить о возникающих при этом неудобствах и ограничениях.

Кроме того, есть два замечания к самой системе MDBS III: во-первых, отсутствие интегрированной среды, что усложняет процесс создания пользователем прикладной БД, особенно на этапе освоения системы, во-вторых, необходимость инициализации БД в полном объеме, объявленном при описании ее логической структуры.

Подводя общий итог, можно сказать, что MDBS III является достаточно простой, элегантной СУБД, обладающей возможностью очень эффективной реализации, а главное, пригодной для построения очень сложных баз данных, ориентированной на программиста-профессионала. Поэтому если Вы желаете создать конкурентоспособный коммерческий продукт в области со сложно структурированными данными, то лучшим вариантом для Вас будет использование расширенной сетевой СУБД MDBS III. Если же Вы пользователь, решающий задачи в области с простой логической структурой данных, то лучше остановиться на реляционных СУБД, так как их возможностей Вам вполне хватит. Теперь Вам остается только точно определить характер стоящей перед вами задачи и выбрать СУБД с соответствующими возможностями. ■

ELITSoft

Пользователям вычислительной техники Акционерное общество "ЭЛИТ СОФТ" предлагает

Агрегаты Бесперебойного Питания

(Uninterruptable Power Source)

Основные характеристики

- Мощность 900 ВА
- Время автономной работы 40 мин.
- Время переключения 5-7 мсек.
- Рабочий диапазон 180-240 В
- Гарантия 1 год
- Послегарантийное обслуживание

В конструкции предусмотрены

- Фильтрация сетевых помех
- Звуковая и световая сигнализация
- Индикация перегрузки
- Индикация уровня заряда батарей
- Герметичные, необслуживаемые аккумуляторные батареи

Поставка и оплата

Поставка со складов в Москве, Санкт-Петербурге, Чите, Рязани.
Любая удобная Вам форма оплаты.

Адрес: 105215 Москва, 11-я Парковая, 44, корп. 1

Телефон:
секретариат: (095) 965-35-29, 532-96-14
отдел сбыта: (095) 965-35-28, 965-35-30
Факс (095) 965-35-28, 965-35-30
Телетайп 205211 КТУР

Возможна доукомплектация компьютеров

Листинг 1

```
/* Пример описания логической структуры БД */
/* в терминах языка описания данных MDBS III*/

/* Раздел описания базы данных      */
db KADRY                          // Имя базы данных
     file "c:NCARD.DB"            // Имя файла на диске
     size 50 pages, page size 512 // Размер файла
     title "Личная карточка"      // Заголовок

/*   Раздел описания пользователей БД   */

user KAREW with KA                // Имя пользователя и пароль доступа
user IVANOV with GB               // Имя пользователя и пароль доступа

/* Раздел описания дополнительных областей БД  */
area BIOGR                        // Имя области
     file "a:biogr.DB"            // Имя файла
     size 20 pages, page size 512 // Размер файла
     title "Биография"            // Заголовок


/*   Раздел описания типов записей  */

record EMPLOYEE                   // Имя записи
       within KADRY               // Имя области
item NUMBER      int 2            // Имя поля, тип и размер в байтах
item NAME        str 20
item FNAME       str 20
item PASTJOB     str 25
item YTDEARN     idec 7,2
       encrypted                  // Установка шифрования поля
       range 0.00 94000.00        //Установка пределов
                                  //допустимых значений
                                  //поля YTDEARN
record DEPT
       within KADRY
     item DNUMBER     int 1
     item NAME        char 12
     item LOCATION    str 35

record ORDER
       within KADRY
     item ONUMBER     int 2
     item DATA        date
     item TEXT        str 80

record JOB
       within KADRY
     item JOBCODE     unsigned 1
     item DESKRIPT    str 20

record SKILL
       within KADRY
     item SKILLCODE   unsigned 2
     item DESCRIPT    str 40
     item RATING      real 2

record SPECIAL
       within KADRY
     item SPECIES     str 15

record BIOGRAPH
       within BIOGR
     item LINE        str 100



/*   Раздел описания наборов    */

set   DETAILS, type 1:1           // Имя набора и его тип
      owner EMPLOYEE              // Имя записи владельца
      member BIOGRAPH             // Имя записи члена

set EXECUT, type 1:n
      owner JOB
      member EMPLOYEE, order sorted ascending (FNAME,NAME)
// EMPLOYEE сортируется по возрастанию (по алфавиту)
// Ключ сортировки состоит из полей FNAME и NAME

set HAS, type 1:n
      owner DEPT
      member EMPLOYEE, order sorted ascending (FNAME,NAME)

set NEEDS, type n:m
      owner JOB, order sorted ascending JOBCODE
      member SKILL, order sorted ascending (RATING,SKILLCODE)

set POSSESS, type n:m
      owner EMPLOYEE, order sorted ascending (FNAME,NAME)
      member SKILL, order sorted ascending SKILLCODE
set IDEP, type 1:n
      owner SYSTEM
      member DEPT, order sorted ascending DNUMBER

set IJOB, type 1:n
      owner SYSTEM
      member JOB, order sorted ascending JOBCODE

set IEMP, type 1:n
      owner SYSTEM
      member EMPLOYEE, order sorted ascending (FNAME,NAME)

set SYS_SPEC, type 1:n
      owner SYSTEM
      member SPECIAL, order sorted ascending SPECIES

set SPEC, type 1:n
      owner EMPLOYEE
      member SPECIAL, order sorted ascending SPECIES

set ARMY, type 1:n
      owner SYSTEM
      member EMPLOYEE, order sorted ascending (FNAME,NAME)

set ORDER_M, type 1:n
      owner ORDER
      member EMPLOYEE, order sorted ascending (FNAME,NAME)

set MANAGES, type 1:n
      owner EMPLOYEE
      member EMPLOYEE, order sorted ascending (FNAME,NAME)
end  // Оператор конца
```

Листинг 2

```
#include "mdbssac.h"
#include "mdbsfunc.h"

struct TEXT_ITEM
{
     char set_name[9];       // Имя набора
     char item_name[9];      // Имя поля в записи
     int mode;               // Режим типа записи в наборе :
} text;                      //   0 - член набора,
                             //   1 - владелец набора

int err;                     // Статус, возвращаемый командами MDBS:
                             //   0 - успешное выполнение команды,
                             //   ? - код ошибки

int List_First(TEXT_ITEM &text)
{
if(!text.mode)
     err = FFM(text.set_name);    // Найти первый член набора,
else                              // связанный с текущим владельцем
     err = FFO(text.set_name);    // Найти первого владельца набора,
return(err);                      // связанного с текущим членом
}

int List_Last(TEXT_ITEM &text)
{
if(!text.mode)
     err = FLM(text.set_name);    // Найти последний член набора,
else                              // связанный с текущим владельцем
     err = FLO(text.set_name);    // Найти последнего владельца набора,
return(err);                      // связанного с текущим членом
}

int List_Next(TEXT_ITEM &text)
{
if(!text.mode)
{ if ((    err = FNM(text.set_name)))  //Найти следующий член набора,
           err = FLM(text.set_name);   // связанный с текущим владельцем
}
else
{ if ((    err = FNO(text.set_name)))  // Найти следующего владельца
           err = FLO(text.set_name);   // набора, связанного с
}                                      // текущим членом
return(err);
}

int List_Prev(TEXT_ITEM &text)
{
if(!text.mode)
{ if ((    err = FPM(text.set_name)))  //Найти предыдущий член набора,
           err = FFM(text.set_name);   //связанный с текущим владельцем
}
else
{ if ((    err = FPO(text.set_name)))  // Найти предыдущего владельца
           err = FFO(text.set_name);   // набора, связанного с текущим
}                                      // членом
return(err);
}

int List_Del(TEXT_ITEM &text)
{
if(!text.mode)
     err = DRM(text.set_name);    // Удалить текущий член набора
else
     err = DRO(text.set_name);    // Удалить текущего владельца набора
return(err);
}
```

Программа для работы с библиотеками процедур PAL

А. Ермолов

Как известно, отлаженные процедуры на встроенном языке СУБД PARADOX лучше хранить и использовать не в символьном виде, а в составе библиотек специального формата, поддерживаемого системой PARADOX. Однако ни в составе самого PARADOX, ни в составе его программатора PPROG, к сожалению, нет никаких средств интерактивной работы с такими библиотеками. На практике это означает, что для создания каждой библиотеки и ее наполнения приходится отдельно писать программу на PAL. PPROG именно так и делает.

Чтобы сделать работу с библиотеками достаточно удобной, я написал на PAL программу lib — библиотекарь, текст которой приведен в этой статье (листинг). Конечно, в приведенной программе можно еще многое усовершенствовать (особенно в плане обработки нештатных ситуаций), однако программа завершена и работать с ней лучше, чем без нее.

Программа lib позволяет выполнять все основные операции с библиотеками: создание, чтение каталога, добавление и удаление модулей.

Кроме того, в меню программы добавлены еще две функции: задание текущей библиотеки и упаковка. Первая функция (Library) — выбор текущей библиотеки — показалась мне удобной, так как я, как правило, за один сеанс выполняю несколько операций с одной библиотекой и имя этой библиотеки удобнее задать один раз в начале сеанса работы. Если Вы хотите работать с новой библиотекой, то указывать ее имя при помощи пункта меню Library не надо: после создания (пункт меню Create) библиотека сразу становится "текущей". Можно также создать новую библиотеку, пользуясь пунктом меню Library — для этого просто надо задать имя несуществующей библиотеки и утвердительно ответить на вопрос программы, хотите ли Вы действительно создать эту библиотеку.

Еще одна функция — упаковка (Pack) нужна потому, что PARADOX не умеет удалять модули из библиотек, в том числе их старые версии. Когда в библиотеку записывается новый модуль под уже существующим именем, то старый не удаляется из библиотеки, а просто становится недоступным. В результате после каждой такой замены модуля библиотека растет. При упаковке все модули библиотеки считываются в память, затем библиотека пересоздается заново (после создания она пустая) и, наконец, в нее добавляются все прочитанные модули. Аналогичным образом производится и удаление (пункт Delete), только удаляемый модуль, конечно, в новую библиотеку не записывается. Кстати, попытка удаления несуществующего модуля эквивалентна упаковке.

Операция добавления новых модулей, как видно из текста, реализуется посредством команды EXECUTE, то есть с помощью создания временного сценария. Это, конечно, неэффективно. Но команда записи модуля в библиотеку — WRITELIB, к сожалению, не может принять переменную в качестве имени библиотеки. Поэтому чтобы указать вводимое пользователем имя модуля, пришлось динамически создавать саму команду записи.

Я уже говорил, что опустил некоторые очевидные усовершенствования, чтобы не загромождать приведенный в данной статье текст. Но одну вещь я не сделал, потому что не смог. Я не смог вывести список процедур, содержащихся в прочитанном файле процедур. Если кто-либо умеет это делать, огромная просьба связаться со мной через журнал "Монитор". Пока в программе применяется нехитрое предположение, что имя процедуры совпадает с именем файла, в котором она находится. Но как быть, если в файле несколько процедур? Тогда их приходится вводить по памяти вручную.

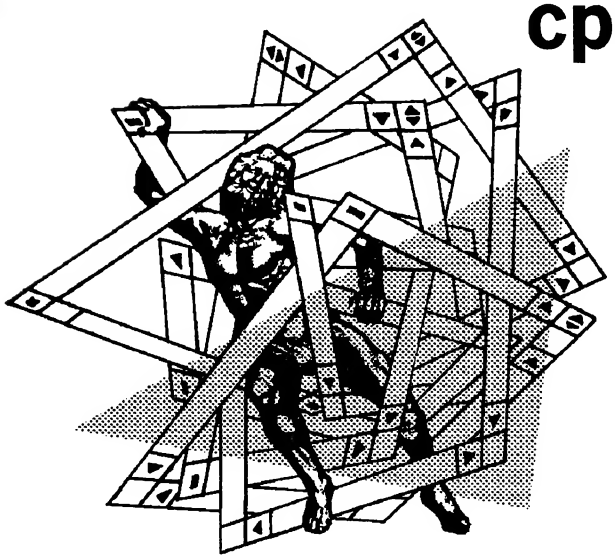
И еще ограничение, которое я не смог преодолеть. Для загрузки текстового файла мне пришлось использовать команду PLAY — ту же, что и для запуска. Если в файле только процедуры и нет главной программы — все хорошо. Но когда я по ошибке попытался добавить в библиотеку процедуры из программы lib, эффект был потрясающий: запускался второй экземпляр скрипта lib поверх первого.

Вот все, что я имел сказать по поводу приведенной программы. Надеюсь, что она принесет кому-нибудь из читателей пользу. ■

Листинг

```
;
;     Библиотекарь для библиотек PARADOX
;
;     (C) Intechcentre LTD
;     А. Ермолов
;
;     Функции:
; - создание библиотеки;
; - добавление скрипта в библиотеку;
; - выдача оглавления библиотеки;
; - удаление (с упаковкой);
; - упаковка (пересоздание).
;
; Переменные: lib- имя библиотеки;
;             modul - имя модуля (при операции добавления);
;             file - файл сценариев при операции добавления (по
;                  умолчанию совпадает с 8 первыми буквами сценария);
;             cur_mod - имя текущего записываемого в библиотеку
;                       модуля при операциях удаления и упаковки;
;
;             choice - выбор элемента меню;
;             ans - буфер ответа на вопрос, задаваемый программой.
;
; Процедура ввода символа с очисткой буфера
PROC GETNEW ()
     WHILE CHARWAITING ()     ; Очистка буфера ввода от мусора
     ENDWHILE
     RETURN GETCHAR ()
ENDPROC
; Проверка того, что имя библиотеки было задано
PROC ISNAME ()
     PRIVATE b                ; Для ввода символа
     CLEAR
     IF lib = "" THEN         ; Библиотека не задана
          @0, 0
          ? "Input Library name (with Librarl or Create item)"
          ? "To continue press any key"
          b = GETNEW ()
          CLEAR
          RETURN False
     ELSE
          RETURN True
     ENDIF
ENDPROC
;
;     ИСПОЛНЯЕМАЯ ПРОГРАММА
;
; Инициализация всех имен пустыми строками
lib = ""
modul = ""
file = ""
WHILE True
; Приглашение к вводу
     SHOWMENU
          "Library":"Name of curent Library(8 chars)",
          "Create":"Create library",
          "Lookup":"Lookup library list",
          "Add":"Add new script to library",
          "Delete":"Delete script and pack library",
          "Pack":"Recreate Library. Delete old versions in Library",
          "Leave":"Exit this script"
          DEFAULT "Library"
     TO choice
     IF NOT Retval OR choice = "Leave" THEN    ; Выход по Esc и по выбору
          QUITLOOP
     ENDIF
; Отработка выбора
     SWITCH
     CASE choice = "Library":
          CLEAR
          @0,0
          ? "Library Name (without extension): "
          ACCEPT "A8" TO lib                   ; Ввод имени библиотеки
          CLEAR
          IF Retval AND LEN (lib) = 0 THEN     ; Выбор имени из меню
               SHOWFILES NOEXT     "*.lib"
                                   "Search Library name"
               TO lib
          ENDIF
          CLEAR
; Проверка наличия библиотеки и создание ее
          IF NOT ISFILE (lib+".lib") THEN      ; Библиотеки нет
               @ 0, 0
               ? "Library "+lib+" not exist. Create (Y/N)?"
               ans = GETNEW ()
               IF ans = "Y" OR ans = "y" THEN
                    CREATELIB lib              ; Создание
               ENDIF
          ENDIF
     CASE choice = "Create":
          CLEAR
     ; Запрос имени библиотеки
          @ 0, 0
          ? "Library Name (without extension): "
          ACCEPT "A8" TO lib                   ; Ввод имени библиотеки
          CLEAR
          IF Retval AND LEN (lib) = 0 THEN     ; Имя не ввели
               MESSAGE "В данном контексте ввод имени обязателен"
               BEEP
          ELSE
               IF Retval THEN
                    ans = "Y"; Признак необходимости создания библиотеки
               IF ISFILE (lib+".lib") THEN     ; Библиотека уже есть
                    CLEAR
                    @ 0, 0
                    ? "This Library already exist. Delete (Y/N)?"
                    ans = GETNEW ()
               ENDIF
               IF ans = "Y" OR ans = "y" THEN  ; Создаем
                    CREATELIB lib
               ENDIF
          ENDIF
     ENDIF
CASE choice = "Lookup":
; Работа всегда с библиотекой lib, а ее существование обеспечено выше
     IF ISNAME () THEN
; Получение листинга библиотеки
          INFOLIB lib
          ; Просмотр до нажатия Esc
          WAIT TABLE
               PROMPT "To continue press Esc"
               UNTIL "Esc"
          CLEARALL
          CLEAR
     ENDIF
CASE choice = "Add":
; Проверка имени
     IF ISNAME () THEN                    ; Библиотека не задана
; Ввод имени файла сценариев
          @0,0
          ? "Input Script File (without extention) "
          ACCEPT "A8" TO file              ; Ввод имени файла сценариев
          CLEAR
          IF Retval AND LEN (file) = 0 THEN     ; Выбор имени из меню
               SHOWFILES NOEXT     "*.sc"
                                   "Search Script File"
               TO file
          ENDIF
          CLEAR
          IF NOT Retval THEN               ; Отказ от ввода
               LOOP
          ENDIF
          PLAY file ; Загрузка
; Ввод имени процедуры (тут, к сожалению, не удастся выбирать из списка)
          @0, 0
          ? "Input adding Modules list: "
          ACCEPT "A40" TO modul
          IF NOT Retval THEN ; Отказ от ввода
               LOOP
          ENDIF
          IF modul = "" THEN               ; По умолчанию имя файла
               modul = file
          ENDIF
; Передача списка в качестве параметра через временный скрипт
          EXECUTE "WRITELIB lib "+modul
     ENDIF
CASE choice = "Delete":
; Ввод имени удаляемого модуля
     @0, 0
     ? "Input Module name: "
     ACCEPT "A40" TO modul
     IF NOT Retval THEN                    ; Отказ от ввода
          LOOP
     ENDIF
; Получение листинга библиотеки
     IF ISNAME () THEN
          ECHO OFF
          INFOLIB lib
          CLEARIMAGE
          ; Чтение
          EXECUTE "READLIB lib ALL"
          ; Пересоздание
          CREATELIB lib
          ; Запись обратно всего, кроме введенного модуля
          VIEW "List"
; Запись модулей в библиотеку
          SCAN
               cur_mod = [Procedure]
               IF NOT (cur_mod = modul) THEN
                    EXECUTE "WRITELIB lib "+cur_mod
               ENDIF
          ENDSCAN
          CLEARIMAGE
     ENDIF
CASE choice = "Pack":
; Получение ее листинга (чтобы знать, что записывать после ее очистки)
     IF ISNAME () THEN
          ECHO OFF
          INFOLIB lib
          CLEARIMAGE
          ; Чтение
          EXECUTE "READLIB lib ALL"
          ; Пересоздание
          CREATELIB lib
          ; Запись обратно
          VIEW "List"
; Запись модулей в библиотеку
          SCAN
               cur_mod = [Procedure]
               EXECUTE "WRITELIB lib "+cur_mod
          ENDSCAN
          CLEARIMAGE
     ENDIF
CASE choice = "Leave":
     OTHERWISE:BEEP
     ENDSWITCH
ENDWHILE
```

Многооконный интерфейс в среде MS Windows



Д. Пласт...

В версиях Windows 3.0 и Windows 3.1 существуют средства поддержки многооконного интерфейса приложения Multiple Document Interface (MDI), описание которого дается в [1]. В этой статье будут даны некоторые разъяснения по организации MDI, необходимые при разработке задач, связанных с использованием многооконного интерфейса.

Окна

MDI используется при создании пользовательского интерфейса для представления и обработки документов различных форм внутри одного приложения. Приложение должно иметь главное окно (frame window), которое во многом схоже с обыкновенным окном, созданным в стиле WS_OVERLAPPEDWINDOW: оно имеет заголовок, меню, пиктограмму системного меню, рамку и кнопки максимизации и минимизации (рис. 1).

В меню различают пункты (items), активизация каждого из которых вызывает появление на экране соответствующего всплывающего (pop-up) меню со своими пунктами.

Оконная процедура главного окна создает так называемое рабочее окно (client window), занимающее всю рабочую область главного окна. Класс рабочего окна MDICLIENT, а следовательно, и его оконная процедура предопределены Windows и служат для стандартизации процесса управления окнами документов. Для этого в Windows 3.0 определены 11 новых сообщений, начинающихся с префикса WM_MDI. Описание работы этих сообщений приведено в конце статьи. Таким образом, не надо регистрировать класс рабочего окна и писать его оконную процедуру.

Перед созданием рабочего окна необходимо заполнить структуру CLIENTCREATESTRUCT, которая определена в windows.h, как показано на рис. 2. Указатель на эту структуру должен быть передан в параметре lParam функции CreateWindow.

Окна документов или дочерние окна (child windows) создаются оконной процедурой главного окна

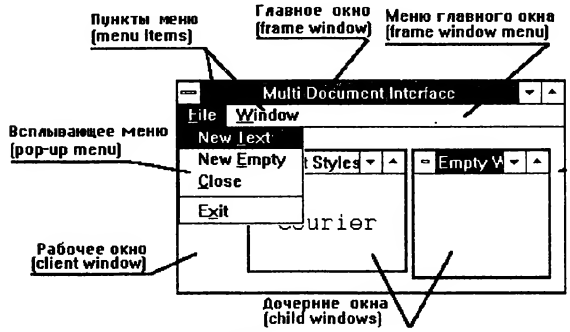


Рис. 1. Структура окна MDI

```
typedef struct tagCLIENTCREATESTRUCT {
    HMENU hWindowMenu;
    WORD  idFirstChild;
}CLIENTCREATESTRUCT;

hWindowMenu  - дескриптор пункта Window в меню главного окна.
               Программа может получить его, используя функцию
               GetSubMenu. (Подробнее о пункте Window см. в
               разделе о дочерних окнах).

idFirstChild - идентификатор первого созданного дочернего окна
               (его порядковый номер в списке открытых окон). Для
               каждого вновь созданного окна Windows увеличивает
               его идентификатор на единицу, а при удалении окон
               переназначает идентификаторы с тем, чтобы они
               образовывали непрерывный числовой ряд. Эти
               идентификаторы используются в сообщениях
               WM_COMMAND к главному окну, когда дочернее окно
               выбирается из всплывающего меню пункта Window, и
               поэтому не должны конфликтовать с другими
               идентификаторами.
```

Рис. 2. Структура CLIENTCREATESTRUCT

путем посылки сообщения WM_MDICREATE рабочему окну, которое, собственно, и создает дочерние окна. Перед вызовом функции SendMessage необходимо заполнить структуру MDICREATESTRUCT (рис. 3).

```
typedef tagMDICREATESTRUCT { LPSTR szClass;
                             LPSTR szTitle;
                             HANDLE hOwner;
                             int x;
                             int y;
                             int cx;
                             int cy;
                             LONG style;
                             LONG lParam;
}MDICREATESTRUCT;

szClass - дальний указатель на строку, содержащую имя дочернего
          класса.
szTitle - дальний указатель на строку, содержащую заголовок окна.
hOwner  - идентификатор экземпляра программы, создавшей дочернее
          окно MDI.
x       - начальная позиция левой стороны окна.
y       - начальная позиция верхней стороны окна.
cx      - начальная ширина окна.
cy      - начальная длина окна.
style - определяет дополнительные типы дочерних окон MDI. Может
принимать одно или несколько из диапазона значений:

WS_MINIMIZE - окно создается в виде пиктограммы;
WS_MAXIMIZE - окно создается распахнутым на всю рабочую область
              главного окна;
WS_HSCROLL  - окно создается с горизонтальной линейкой прокрутки;
WS_VSCROLL  - окно создается с вертикальной линейкой прокрутки.
lParam  - 32-битовое значение, определяемое приложением.
```

Рис. 3. Структура MDICREATESTRUCT

К дочерним окнам MDI стандартом CUA предъявляются следующие требования.

1. Дочернее окно может быть минимизировано. При этом его пиктограмма возникает в нижней части рабочей области главного окна.

2. Дочернее окно может быть максимизировано, то есть распахнуто до размеров рабочей области главного окна. В этом случае его заголовок (обычно используемый для показа имени файла, в котором хранится документ) исчезает и добавляется к заголовку главного окна. Первым пунктом меню главного окна становится кнопка системного меню дочернего окна, а последним пунктом — кнопка максимизации/минимизации дочернего окна.

3. В акселераторах клавиатуры дочерних окон должна использоваться клавиша Ctrl (а не Alt, как у главного окна). Стандартно определены следующие комбинации клавиш:

- Ctrl-F4 — закрыть окно;
- Ctrl-F6 — перейти к следующему окну.

Однако существуют и акселераторы с Alt:

- Alt-–(минус) — активизировать системное меню дочернего окна;
- Alt-Spacebar — активизировать системное меню главного окна.

4. При использовании клавиш "стрелка влево"/"стрелка вправо" для передвижения по пунктам меню управление передается следующим образом: системное меню главного окна —> системное меню активного дочернего окна —> пункты меню главного окна.

5. Меню главного окна всегда имеет пункт под названием Window (тот самый, дескриптор которого мы описывали в структуре CLIENTCREATESTRUCT). Этот пункт должен быть последним, за исключением пункта Help, если таковой существует. Пункт Window содержит всплывающее меню, управляющее отображением дочерних окон в пределах рабочей области MDI. Помимо этого, в данном подменю помещается список всех открытых окон документов с их идентификаторами (параметр idFirstChild в структуре CLIENTCREATESTRUCT) и помечается активный документ. Из этого списка можно активизировать другой документ; при этом его окно переместится на передний план.

6. В случае подгрузки документов, требующих различных методов обработки, дочерние окна могут изменять меню главного окна согласно требованиям, предъявляемым к этим методам. Изменение производится посылкой сообщения WM_MDISETMENU рабочему окну. Может быть изменено также и количество пунктов меню; при этом необходимо помнить о позиции подменю Window, которое должно быть последним.

Общую иерархию окон в MDI можно представить следующим образом:

```
                                  |—> дочернее окно
главное окно —> рабочее окно ——   |—> ...
                                  |—> дочернее окно
```

Реализация

Рассмотрим работу MDI на простом примере. Программа SMDI, приведенная в листингах 1-7, подгружает два вида документов. В окно первого документа с заголовком Text выводится название текущего шрифта, причем пользователь имеет возможность изменять этот шрифт индивидуально для каждого окна. Второй документ, Empty, выводит пустое окно и служит лишь для иллюстрации того, как изменяется меню главного окна при переходе от одного документа к другому.

Функция WinMain

Функция WinMain, приведенная в листинге 1, регистрирует класс главного окна MDIFrame и все классы дочерних окон MDITextChild и MDIEmptyChild.

В классе MDITextChild параметр cbWndExtra содержит размер локального дескриптора. В оконной процедуре данного класса этот дескриптор будет определять структуру, содержащую информацию, уникальную для каждого экземпляра класса.

Все классы регистрируются со стандартной пиктограммой и стандартной формой курсора.

Далее получаются дескрипторы трех различных меню, на работе с которыми есть смысл остановиться подробнее. Как уже было сказано выше, только главное окно MDI имеет свое меню. Однако дочерние окна могут его изменять. Рассмотрим этот процесс.

В работе нашего MDI можно различить три состояния. Первое возникает при инициализации MDI, когда на экран выведены только главное и рабочее окна. При этом должно быть активизировано меню главного окна MDIMenuInit, которое описано в файле ресурсов SMDI.RS (листинг 6). Меню содержит один пункт File, позволяющий подгружать и закрывать окна документов, а также выходить из программы.

Второе состояние MDI возникает при подгрузке документа класса MDITextChild. При этом активизируется другое меню главного окна — MDIMenuText. В нем три пункта: File, идентичный пункту File меню MDIMenuInit, Text — для выбора гарнитуры шрифта текущего документа и Window — стандартный пункт для манипуляции отображением на экране дочерних окон. Пункт Window содержит также список всех активных дочерних окон с указателем на текущее активное.

Третье состояние наступает, когда подгружен документ класса MDIEmptyChild. Новое меню MDIMenuEmpty содержит два пункта: File и Window.

Для своевременной активизации нужного меню программе необходимо подгрузить все меню и сохранить их дескрипторы в глобальных переменных, определенных в заголовочном файле SMDI.H (листинг 5).

Следует обратить особое внимание на поведение пункта Window. При первом состоянии MDI его нет совсем, так как ни одно дочернее окно не подгружено и манипулировать нечем. Во втором состоянии пункт Window является вторым по счету пунктом меню главного окна (нумерация ведется от нулевого значения), а в третьем состоянии — первым пунктом меню. Заметим также, что список открытых документов, содержащийся в пункте Window, автоматически присоединяется к всплывающему меню этого пункта при инициализации соответствующего меню. При этом MS Windows необходимо знать дескриптор пункта Window в каждом меню для того, чтобы правильно присоединять список документов. Естественно, что для каждого из подгружаемых меню этот дескриптор уникален.

Все дескрипторы пункта Window могут быть получены при помощи функции GetSubMenu, где первым параметром идет дескриптор меню главного окна, а вторым — порядковый номер пункта Window в этом меню. Все номера определены в SMDI.H. И если с меню MDIMenuText и MDIMenuEmpty все ясно: пункт Window присутствует в них в соответствующих позициях, — то в меню MDIMenuInit пункта Window нет. Тем не менее, мы инициализируем дескриптор этого пункта, приписывая ему нулевую позицию в меню. Согласно сказанному ранее, список всех открытых дочерних окон должен быть присоединен к пункту File этого меню. Однако такого не произойдет, ибо при активизации меню MDIMenuInit нет ни одного открытого документа. Поэтому пункт Window в данном случае можно ассоциировать с любым пунктом меню главного окна.

Дескриптор подгружаемой далее таблицы клавиш-акселераторов, определенной в файле ресурсов SMDI.RS (листинг 6), является локальным для функции WinMain и впоследствии используется в функции TranslateAccelerator.

После получения дескрипторов всех ресурсов необходимо создать главное окно приложения и передать ему в качестве параметра дескриптор меню MDIMenuInit. Дескриптор рабочего окна можно получить с помощью функции GetWindow, зная, что оно является первым дочерним окном главного окна.

Цикл сообщений для MDI имеет более сложную форму. В него включена функция TranslateMDISysAccel, переводящая сообщение WM_KEYDOWN в сообщение WM_SYSCOMMAND для активного дочернего окна. Эта функция возвращает FALSE, если данное сообщение не является сообщением клавиш-акселераторов системного меню дочернего окна; в этом случае приложение использует функцию TranslateAccelerator для определения, является ли данная комбинация клавиш акселератором, определенным для данного приложения. Если сообщения от акселераторов обработаны этими двумя функциями, их не надо посылать для обработки функции TranslateMessage.

Оконная процедура главного окна

Оконная процедура главного окна (листинг 2) создает рабочее окно и посылает ему сообщения для управления дочерними окнами.

По сообщению WM_CREATE заполняется структура CLIENTCREATESTRUCT, параметр hWindowMenu которой является дескриптором пункта Window из меню "MDIMenuInit", полученным нами в функции WinMain. Указатель на структуру CLIENTCREATESTRUCT передается в качестве параметра функции CreateWindow при создании рабочего окна.

Сообщение WM_COMMAND в параметре wParam содержит идентификатор выбранного пункта всплывающего меню (идентификаторы с префиксом IDM_, соответствующие пунктам меню, определены в SMDI.H).

В случае выбора открытия новых документов IDM_NEWTEXT и IDM_NEWEMPTY заполняется структура MDICREATESTRUCT и рабочему окну посылается сообщение WM_MDICREATE для создания дочернего окна. Указатель на структуру MDICREATESTRUCT, соответствующим образом преобразованный к типу LONG, передается в качестве последнего параметра функции SendMessage.

При обработке пунктов всплывающего меню, определяющих способ отображения дочерних окон на рабочей области главного окна (IDM_TILE, IDM_CASCADE, IDM_ARRANGE), соответствующие сообщения с префиксом WM_MDI посылаются рабочему окну.

При закрытии активного дочернего окна IDM_CLOSE сначала определяется его дескриптор путем посылки сообщения WM_MDIGETACTIVE рабочему окну. Далее посылается запрос дочернему окну, согласно ли оно на закрытие (сообщение WM_QUERYENDSESSION), и, если ответ удовлетворительный, рабочему окну посылается сообщение WM_MDIDESTROY для закрытия дочернего окна.

Пункт IDM_EXIT служит для выхода из программы. При этом главное окно посылает себе же сообщение WM_CLOSE.

Пункт IDM_CLOSEALL обрабатывается несколько сложнее. Определим экспортируемую функцию CloseEnumProc, служащую для закрытия активного дочернего окна. Происходит это почти так же, как и при выборе пункта IDM_CLOSE. С помощью функции MakeProcInstance получим указатель на точку вызова функции CloseEnumProc. Далее при помощи функции EnumChildWindows перебираем все дочерние окна, передавая их дескрипторы функции CloseEnumProc. Перебор продолжается до тех пор, пока все дочерние окна не будут закрыты. После этого с помощью функции FreeProcInstance необходимо освободить точку вызова функции CloseEnumProc. Отметим также, что функция CloseEnumProc должна быть отмечена как EXPORTS в файле определения модуля SMDI.DEF (листинг 7).

Если параметр wParam сообщения WM_COMMAND не содержит ни одного из перечисленных выше значений, то значит активным меню является не MDIMenuInit, а либо MDIMenuText, либо MDIMenuEmpty, содержащее другие всплывающие меню со своими идентификаторами пунктов. Один из них и содержится в wParam. Поэтому обработку таких идентификаторов необходимо передать оконной процедуре того дочернего класса, к которому принадлежит текущее активное окно документа. Это делается после определения дескриптора активного окна посылкой ему сообщения WM_COMMAND с текущими значениями параметров wParam и lParam.

Сообщение WM_CLOSE приходит к оконной процедуре главного окна после выбора пункта с идентификатором IDM_EXIT. При этом оконная процедура посылает самой себе сообщение WM_COMMAND с параметром wParam, равным IDM_CLOSEALL.

Сообщение WM_DESTROY стандартное и служит для окончания работы с приложением.

Все не обрабатываемые оконной процедурой главного окна сообщения должны быть переданы функции DefFrameProc, заменившей стандартную DefWindowProc. Существует также несколько сообщений, которые должны быть переданы для обработки функции DefFrameProc вне зависимости от того, обрабатываются они оконной процедурой или нет. Это WM_MENUCHAR, WM_NEXTMENU, WM_SETFOCUS и WM_SIZE. Вторым параметром функции DefFrameProc должен быть дескриптор рабочего окна.

Оконная процедура класса MDITextChild

Оконная процедура класса MDITextChild (листинг 3), помимо стандартного кода обработки дочернего окна, содержит код для вывода в центре окна строки текста с названием шрифта, который пользователь выбирает из меню MDIMenuText. Выбранный шрифт становится текущим, и его название выводится этим же шрифтом. Рассмотрение работы со шрифтами не входит в задачу данной статьи; более подробную информацию можно почерпнуть из [1],[5]. Мы остановимся лишь на аспектах, связанных с MDI.

Когда оконной процедуре рабочего окна приходит сообщение WM_MDICREATE, она создает дочернее окно и посылает ему сообщение WM_CREATE. Процедура дочернего окна использует это сообщение в следующих целях. Во-первых, необходимо инициализировать структуру данных, уникальных для каждого экземпляра окна, если таковая структура предусмотрена и под ее дескриптор отведено место в параметре cbWndExtra при регистрации класса окна. В нашем случае эта структура называется TEXTDATA и определена она в файле SMDI.H (листинг 5). Первый элемент структуры, nItem, является идентификатором выбранного пункта всплывающего меню Text; второй, logFnt, — структурой типа LOGFONT для задания логического шрифта. Поля структуры TEXTDATA заполняются значениями по умолчанию, и дескриптор ее передается функции SetWindowWord для изменения атрибута окна. Во-вторых, с помощью функции GetParent необходимо определить дескрипторы главного и рабочего окон.

Параметр wParam сообщения WM_COMMAND может содержать идентификаторы только тех пунктов всплывающих меню, которые не были обработаны оконной процедурой главного окна. В данном случае эти пункты и всплывающее меню, их содержащее, принадлежат меню MDIMenuText, и параметр wParam принимает следующие значения: IDM_ROMAN, IDM_SWISS, IDM_MODERN, IDM_SCRIPT и IDM_DECORATIVE. Все они описаны в SMDI.H. При выборе любого из этих пунктов процедура окна получает дескриптор меню и с помощью функции CheckMenuItem снимает отметку с предыдущего выбранного пункта и ставит ее на вновь выбранный пункт. При этом реинициализируются поля структуры LOGFONT, необходимые для задания шрифта, соответствующего выбранному пункту всплывающего меню. Значения для реинициализации берутся из заранее заданных статических массивов.

При получении сообщения WM_PAINT происходит установка нового логического шрифта и вывод его названия в центр окна. При этом функция DrawText используется дважды: первый раз для затирания предыдущего текста, а второй — для вывода новой строки.

Сообщение WM_MDIACTIVATE приходит к дочернему окну, когда оно должно быть либо активизировано, либо деактивизировано в зависимости от значения параметра wParam. Если дочернее окно активизируется, рабочему окну посылается сообщение WM_MDISETMENU на установку меню, соответствующего классу данного дочернего окна. В противном случае устанавливается меню MDIMenuInit, соответствующее классу главного окна приложения. После этого область меню должна быть обязательно перерисована функцией DrawMenuBar.

Сообщения WM_CLOSE и WM_QUERYENDSESSION вызывают появление на экране окна, созданного с помощью функции MessageBox, с подтверждением на закрытие окна.

По приходе сообщения WM_DESTROY происходит высвобождение блока локальной памяти, который был распределен под структуру TEXTDATA.

Все необработанные сообщения посылаются функции DefMDIChildProc, используемой вместо DefWindowProc. Аналогично оконной процедуре главного окна, следующие сообщения в любом случае должны быть переданы функции DefMDIChildProc для обработки: WM_CHILDACTIVATE, WM_MENUCHAR, WM_GETMINMAXINFO, WM_MOVE, WM_SETFOCUS, WM_SIZE и WM_SYSCOMMAND.

Оконная процедура класса MDIEmptyChild

Класс MDIEmptyChild создан для иллюстрации изменения меню главного окна приложения при активизации окон различных классов (листинг 4). Оконная процедура этого класса при получении сообщения WM_CREATE определяет дескрипторы рабочего и главного окон, а при получении сообщения WM_MDIACTIVATE устанавливает либо меню MDIMenuEmpty, соответствующее своему классу, либо меню MDIMenuInit, соответствующее классу главного окна приложения.

Заключение

Конечно, если Вы программируете на Turbo C++ 3.0 for Windows фирмы Borland International, то Вы, вероятнее всего, используете Object Windows, впрочем, так делаю и я. Однако сокрытие информации, являющееся одной из основных черт объектно-ориентированного программирования, зачастую не позволяет до конца понять суть того, что же происходит с программой. Надеюсь, что изложенный материал будет Вам полезен при написании программ как на C, так и на C++.

К сожалению, в статье с ограниченным объемом невозможно отразить все нюансы организации многооконного интерфейса. К тому же, все аспекты программирования под MS Windows тесно переплетены между собой и требуют настойчивого и комплексного изучения. Поэтому хотелось бы предложить список литературы, которая может быть полезна в работе. ■

Список рекомендуемой литературы

1. Microsoft Windows Software Development Kit. Guide to Programming. Version 3.0.— Microsoft Corporation, 1990.
2. Microsoft Windows Software Development Kit. Reference. Version 3.0.— Microsoft Corporation, 1990.— Vol. 1,2.
3. Petzold Ch. Programming Windows. The Microsoft guide to writing applications for Windows 3.— Microsoft Press, 1990.
4. Гладков С.А.,Фролов Г.В. Программирование в Microsoft Windows: в 2-х частях.— М.: Диалог-МИФИ, 1992.
5. Windows 3.0. Пособие по программированию/Под ред. Рябова А.Ю.— М.: ВНПП "Аналитик", 1992.
6. Windows 3.0. Справочник для программистов/Под ред. Авдеева Е.В.— М.: ППИ НЦ, 1992.

Справочная колонка

Сообщения MDI

Приложение MDI управляет дочерними окнами, посылая рабочему окну специальные сообщения с префиксом WM_MDI. Дадим краткую характеристику этим сообщениям.

WM_MDICREATE

Оконная процедура главного окна посылает это сообщение рабочему окну для создания нового дочернего окна. Параметр wParam при этом не используется, а параметр lParam должен содержать дальний указатель на структуру MDICREATESTRUCT. При этом значение, возвращаемое функцией SendMessage, содержит дескриптор нового дочернего окна в младшем слове и нуль в старшем слове.

Заголовок окна добавляется в список открытых окон во всплывающем меню пункта Window из меню главного окна, окну присваивается уникальный идентификатор, на единицу превышающий идентификатор последнего созданного окна.

Когда дочернее окно создано, Windows посылает ему сообщение WM_CREATE, параметр lParam которого содержит указатель на структуру CREATESTRUCT. Поле lpCreateParams этой структуры содержит дальний указатель на структуру MDICREATESTRUCT, используемую при создании рабочего окна. Рассмотрение структуры CREATESTRUCT не входит в задачу данной статьи.

WM_MDIACTIVATE

Приложение посылает это сообщение рабочему окну, информируя о необходимости активизировать дочернее окно. Рабочее окно обрабатывает это сообщение и посылает его же как активизируемому, так и деактивизируемому окну.

Когда приложение посылает сообщение WM_MDICREATE своему рабочему окну, параметр wParam содержит дескриптор дочернего окна, которое должно быть активизировано, а параметр lParam устанавливается в NULL. Когда же рабочее окно посылает это сообщение дочернему, wParam устанавливается в TRUE, если дочернее окно активизируется, и в FALSE, если оно деактивизируется. При этом старшее слово lParam содержит дескриптор деактивизируемого, а младшее слово — дескриптор активизируемого окна.

WM_MDIDESTROY

Посылается рабочему окну, когда приложение хочет закрыть дочернее окно. При этом параметр lParam не используется, а wParam содержит дескриптор дочернего окна, которое необходимо закрыть. Заголовок окна удаляется из списка открытых окон, а идентификаторы окон переназначаются. Если закрытое дочернее окно не последнее, то активным становится следующее за ним окно.

WM_MDIGETACTIVE

Это сообщение посылается рабочему окну для определения текущего активного дочернего окна. Параметры wParam и lParam не используются. Значение, возвращаемое функцией SendMessage, в младшем слове содержит дескриптор этого окна; старшее слово содержит 1, если окно максимизировано, и 0 — в противном случае.

WM_MDISETMENU

Это сообщение заменяет меню главного окна MDI, всплывающее меню пункта Window или и то и другое. Параметр wParam не используется, а параметр lParam в младшем слове содержит дескриптор меню главного окна, а в старшем — дескриптор пункта Window. Если либо младшее, либо старшее слово установлены в нуль, соответствующее меню не изменяется. При этом значение, возвращаемое функцией SendMessage, содержит в младшем слове дескриптор главного меню приложения.

При замене всплывающего меню пункта Window список открытых дочерних окон изымается из старого всплывающего меню и добавляется в новое.

WM_MDIICONARRANGE

Это сообщение посылается оконной процедурой главного окна приложения рабочему окну для упорядочения всех минимизированных дочерних окон. Оно не влияет на окна, которые не минимизированы. Параметры wParam и lParam не используются.

WM_MDIMAXIMIZE

Это сообщение посылается рабочему окну для максимизации активного дочернего окна до размеров рабочей области главного окна. Кнопка системного меню и кнопка восстановления состояния дочернего окна добавляются в меню главного окна приложения. Параметр lParam не используется; параметр wParam содержит дескриптор дочернего окна.

WM_MDINEXT

Это сообщение активизирует следующее дочернее окно, находящееся сразу за текущим активным окном, и размещает новое активное окно впереди всех остальных. Параметры wParam и lParam не используются.

WM_MDIRESTORE

Это сообщение восстанавливает дочернее окно из минимизированного или максимизированного состояния. Параметры wParam и lParam не используются.

WM_MDICASCADE

Это сообщение переводит все дочерние окна MDI в формат "каскада". Параметры wParam и lParam не используются.

WM_MDITILE

Это сообщение переводит все дочерние окна MDI в формат "плитки". Параметры wParam и lParam не используются. ■

Корпорация
Софткит
Softkit
Corporation

117607, Москва, тел.(095)932-69-13
Мичуринский пр-т, 31/1 (095)931-79-33

Предприятиям, Организациям и Фирмам

Пользователям системы КЛИППЕР

Корпорация Софткит

представляет

ТУЛКИТ для КЛИППЕРа

Инструментальный пакет программ для профессиональных пользователей системы КЛИППЕР

К Л И П К И Т

(C) Корпорация Софткит Версия 1.0, 1.1, 1992 г., Москва

Инструментальный пакет "К л и п к и т" Версия 1.0, 1.1

Представляет собой мощное инструментальное средство для профессиональных программистов - разработчиков прикладных систем и АРМов на базе КЛИППЕР.

Пакет включает библиотеку программ с более чем 500 пользовательскими функциями и рядом сервисных утилит, которые обеспечивают функционально полный набор интерфейсных средств для КЛИППЕРа (Summ'87/5.01), существенно расширяющих и упрощающих использование его базовых возможностей, повышающих качество разрабатываемых прикладных систем.

Пакет включает следующие основные направления средств:

- ОКНА
- МЕНЮ
- СООБЩЕНИЯ
- УСТРОЙСТВА
- МЫШЬ, КУРСОР
- УПРАВЛЕНИЕ ПАМЯТЬЮ
- КЛАССЫ ОБЪЕКТОВ
- ГРАФИКУ
- КОММУНИКАЦИИ
- СЕРВИС

"В ХОРОШИХ ИНСТРУМЕНТАХ НУЖДАЮТСЯ ВСЕ, НО В ПЕРВУЮ ОЧЕРЕДЬ - ВЕЛИКИЕ МУЗЫКАНТЫ"

Страдивари

Инструментальный пакет "К л и п к и т" Версия 1.0, 1.1
Условия поставки

Первая партия пакета в полной комплектации поступит в распространение во втором квартале 1993 г.

Цена пакета - 15 000/30 000 руб.+ НДС.

Распространение пакета осуществляется первоначально по предварительным заказам, а затем в свободной продаже. Оформление предварительного заказа дает право на 20% скидку к цене пакета независимо от количества комплектов в заказе (минимальный объем заказа для организаций - 2 комплекта).

Заказ на приобретение пакета партиями (10 и более штук) дополнительно дает право на следующие скидки к цене пакета:
от 10 до 50 - 10%;
от 50 до 100 - 20%;
от 100 и более - 30%.

По желанию заказчика в комплект поставки может быть включена в соответствии с диллерским договором система Clipper & Clipper Tools One (Summ'87/5.01) по цене Computer Associates (СП "МАГНИТ") со скидкой 10%, включая НДС. При этом указанные выше правила скидок на поставку системы Клиппер не распространяются!

Для оформления предварительного заказа необходимо направить по адресу Корпорации бланк заказа (письмо от организации) с указанием комплектации и количества комплектов, обратного почтового адреса и копии документа о перечислении предоплаты в соответствии с отмеченными выше условиями на первую партию пакета во втором квартале 1993 г.

Приобретение партий пакета для последующего распространения осуществляется только на основе заключенного диллерского договора.

С момента поступления предоплаты на счет заказ считается оформленным. Выполнение заказа может быть осуществлено как на территории Корпорации по предварительным приглашениям, высылаемым по адресу заказчика, так и пересылкой пакета почтой.

Для формирования предварительных заказов и более полного информирования потенциальных пользователей о возможностях пакета и технологии разработки программ с его применением в настоящее время подготовлены и распространяются демонстрационный(DEMO) и учебный(TUTOR) комплекты пакета, являющиеся дополнительными составляющими полной комплектации пакета, распространяемыми независимо.

Приобретение этих комплектов позволяет сразу приступить к практическому освоению технологий использования пакета в прикладных разработках и принять обоснованное решение о целесообразности оформления предварительного заказа на полную версию пакета.

Цена демонстрационного комплекта(DEMO) - 500/1000 руб. + НДС.

Цена учебного комплекта(TUTOR) - 1500/2000 руб. + НДС.

Приобретение этих комплектов дает право на следующие скидки к цене пакета:
демонстрационный комплект(DEMO) - 10%;
учебный комплект(TUTOR) - 20%.

Зарегистрированные пользователи получают право на приобретение всех последующих версий пакета с 25% скидкой к цене пакета, пользоваться консультациями и методической поддержкой специалистов Корпорации.

В состав полной комплектации пакета входят:

1. Документация (250 стр.) содержащая:
   - руководство пользователя;
   - справочное руководство;
   - учебное пособие по приложениям.
2. Лицензионное соглашение пользователя.
3. Регистрационная карточка пользователя.
4. Четыре дистрибутивные дискеты(типа DS/DD,360K) с архивированной информацией (.LZH), включающие:
   Диск#1 - инстал/утилиты;
   Диск#2 - библиотеки;
   Диск#3 - справочник(NG);
   Диск#4 - приложения.

В состав демонстрационного комплекта (Диск DEMO) входят:

1. Документация, которая содержит рекламные проспекты, инструкции, описания примеров, бланки заказа на полный комплект пакета.
2. Одна дистрибутивная дискета (типа DS/DD, 360K) с информацией в архивированном виде (.LZH), содержащей инструкции и программу инсталяции, исполняемые демонстрационные программы (.EXE), исходные тексты программ(.PRG) с комментариями, файлы демонстрационных баз данных и примеров.

В состав учебного комплекта (Диск TUTOR) входят:

1. Документация, которая содержит рекламные проспекты, инструкции, описания примеров, лицензионное соглашение на использование библиотек учебного пакета, бланки заказа на полный комплект пакета.
2. Одна дистрибутивная дискета (типа DS/DD, 360K) с информацией в архивированном виде (.LZH), содержащей инструкции и программу инсталяции, учебную библиотеку пакета с более чем 100 пользовательскими функциями (.LIB), утилиты, исходные тексты демонстрационных программ (.PRG) с комментариями, файлы демонстрационных баз данных и примеров.

ЗАКАЗЫ НАПРАВЛЯТЬ ПО АДРЕСУ: 117607, Москва, Мичуринский пр-т, 31/1, ТМО

БАНКОВСКИЕ РЕКВИЗИТЫ: р/с 006468455 в МАБ "Темпбанк" корр./сч.161104 в РКЦ ГУ ЦБ РФ в г. Москве, МФО 201791, Корпорация СОФТКИТ.

Листинг 1. Функция WinMain

```
//-------------------------------------------------
//--- SMDI.C -- простой многооконный интерфейс
//--- Copyright (c) D.Plastinin 1992

#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include "smdi.h"

//---Глобальные переменные
char    szFrameClass[]    = "MDIFrame";
char    szTextClass[]     = "MDITextChild";
char    szEmptyClass[]    = "MDIEmptyChild";

int PASCAL WinMain(   HANDLE hInstance,
                      HANDLE hPrevInstance,
                      LPSTR lpszCmdLine,
                      int nCmdShow )
{

HANDLE      hAccel;
HWND        hwndFrame, hwndClient;
MSG         msg;
WNDCLASS    wndclass;

hInst = hInstance;      //---Сохранить идентификатор экземпляра

If( !hPrevInstance ) {

//---Зарегистрировать класс главного окна
    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc    = FrameWndProc;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;
    wndclass.hInstance      = hInstance;
    wndclass.hIcon          = LoadIcon( NULL, IDI_APPLICATION );
    wndclass.hCursor        = LoadCursor( NULL, IDC_ARROW );
    wndclass.hbrBackground  = COLOR_APPWORKSPACE + 1;
    wndclass.lpszMenuName   = NULL;
    wndclass.lpszClassName  = szFrameClass;

    RegisterClass( &wndclass );

//---Зарегистрировать класс MDITextChild
    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc    = TextWndProc;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = sizeof( LOCALHANDLE );
    wndclass.hInstance      = hInstance;
    wndclass.hIcon          = LoadIcon( NULL, IDI_APPLICATION );
    wndclass.hCursor        = LoadCursor( NULL, IDC_ARROW );
    wndclass.hbrBackground  = GetStockObject( WHITE_BRUSH );
    wndclass.lpszMenuName   = NULL;
    wndclass.lpszClassName  = szTextClass;

    RegisterClass( &wndclass );

//---Зарегистрировать класс MDIEmptyChild
    wndclass.style          = CS_HREDRAW | CS_VREDRAW;
    wndclass.lpfnWndProc    = EmptyWndProc;
    wndclass.cbClsExtra     = 0;
    wndclass.cbWndExtra     = 0;
    wndclass.hInstance      = hInstance;
    wndclass.hIcon          = LoadIcon( NULL, IDI_APPLICATION );
    wndclass.hCursor        = LoadCursor( NULL, IDC_ARROW );
    wndclass.hbrBackground  = GetStockObject( GRAY_BRUSH );
    wndclass.lpszMenuName   = NULL;
    wndclass.lpszClassName  = szEmptyClass;

    RegisterClass( &wndclass );
}

//---Получить дескрипторы меню главного окна для каждого класса
hMenuInit     = LoadMenu( hInst, "MDIMenuInit" );
hMenuText     = LoadMenu( hInst, "MDIMenuText" );
hMenuEmpty    = LoadMenu( hInst, "MDIMenuEmpty" );

//---Получить дескрипторы пункта Window для каждого класса
hMenuInitWindow     = GetSubMenu( hMenuInit, INIT_MENU_POS );
hMenuTextWindow     = GetSubMenu( hMenuText, TEXT_MENU_POS );
hMenuEmptyWindow    = GetSubMenu( hMenuText, EMPTY_MENU_POS );

//---Загрузить таблицу акселераторов
hAccel = LoadAccelerators( hInst, "MDIAccel" );

//---Создать главное окно
hwndFrame = CreateWindow( szFrameClass,
                  "Multi Document Interface",
                  WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  CW_USEDEFAULT, CW_USEDEFAULT,
                  NULL,
                  hMenuInit,
                  hInstance,
                  NULL
    );

//---Получить дескриптор рабочего окна
hwndClient = GetWindow( hwndFrame, GW_CHILD );

ShowWindow( hwndFrame, nCmdShow );
UpdateWindow( hwndFrame );

//---Измененный цикл сообщений
while(    GetMessage( &msg, NULL, 0, 0 ))
          If(   !TranslateMDISysAccel( hwndClient, &msg ) &&
                !TranslateAccelerator( hwndFrame, hAccel, &msg ))
                {
                TranslateMessage( &msg );
                DispatchMessage( &msg );
                }

return msg.wParam;
}
```

Листинг 2. Оконная процедура главного окна

```
//-------------------------------------------------
//--- FrameWndProc - оконная процедура главного окна
long FAR PASCAL FrameWndProc(     HWND hwnd, WORD message,
                                  WORD wParam, LONG lParam )
{

static HWND hwndClient;
CLIENTCREATESTRUCT clientcreate;
FARPROC lpfnEnum;
HWND hwndChild, hwndNext;
MDICREATESTRUCT mdicreate;

switch( message ) {

    case WM_CREATE:
        clientcreate.hWindowMenu = hMenuInitWindow;
        clientcreate.idFirstChild = IDM_FIRSTCHILD;

        hwndClient = CreateWindow( "MDICLIENT",
                                   NULL,
                                   WS_CHILD | WS_CLIPCHILDREN |
                                   WS_VISIBLE,
                                   0, 0, 0, 0,
                                   hwnd, 1, hInst,
                                   (LPSTR) &clientcreate );
        return 0;

    case WM_COMMAND:
        switch( wParam ) {

        case IDM_NEWTEXT:
            mdicreate.szClass   = szTextClass;
            mdicreate.szTitle   = "Text Styles";
            mdicreate.hOwner    = hInst;
            mdicreate.x         = CW_USEDEFAULT;
            mdicreate.y         = CW_USEDEFAULT;
            mdicreate.cx        = CW_USEDEFAULT;
            mdicreate.cy        = CW_USEDEFAULT;
            mdicreate.style     = 0;
            mdicreate.lParam    = NULL;

            hwndChild = SendMessage(  hwndClient,
                                      WM_MDICREATE,
                                      0,
                                      (LONG)(LPMDICREATESTRUCT)
                                      &mdicreate );
            return 0;

        case IDM_NEWEMPTY:
            mdicreate.szClass   = szEmptyClass;
            mdicreate.szTitle   = "Empty Window";
            mdicreate.hOwner    = hInst;
            mdicreate.x         = CW_USEDEFAULT;
            mdicreate.y         = CW_USEDEFAULT;
            mdicreate.cx        = CW_USEDEFAULT;
            mdicreate.cy        = CW_USEDEFAULT;
            mdicreate.style     = 0;
            mdicreate.lParam    = NULL;

            hwndChild = SendMessage(  hwndClient,
                                      WM_MDICREATE,
                                      0,
                                      (LONG)(LPMDICREATESTRUCT)
                                      &mdicreate );
            return 0;

        case IDM_CLOSE:
            hwndChild = LOWORD( SendMessage( hwndClient,
                                             WM_MDIGETACTIVE,
                                             0, 0L ));
            If(   SendMessage( hwndChild, WM_QUERYENDSESSION,
                               0, 0L ))
                  SendMessage( hwndClient, WM_MDIDESTROY,
                               hwndChild, 0L );
            return 0;

        case IDM_EXIT:
            SendMessage( hwnd, WM_CLOSE, 0, 0L );
            return 0;

        case IDM_TILE:
            SendMessage( hwndClient, WM_MDITILE, 0, 0L );
```

Листинг 2 (продолжение)

```
            return 0;

        case IDM_CASCADE:
            SendMessage( hwndClient, WM_MDICASCADE, 0, 0L );
            return 0;

        case IDM_ARRANGE:
            SendMessage( hwndClient, WM_MDIICONARRANGE, 0, 0L );
            return 0;

        case IDM_CLOSEALL:
            lpfnEnum = MakeProcInstance( (FARPROC) CloseEnumProc,
                                                    hInst );
            EnumChildWindows( hwndClient, lpfnEnum, 0L );
            FreeProcInstance( lpfnEnum );
            return 0;

        default:
            hwndChild = LOWORD( SendMessage(hwndClient,
                                            WM_MDIGETACTIVE,
                                            0, 0L));
            if(  IsWindow( hwndChild ))
                SendMessage( hwndChild, WM_COMMAND,
                             wParam, lParam );
            break;

        }
    break;

    case WM_QUERYENDSESSION:
    case WM_CLOSE:
        SendMessage( hwnd, WM_COMMAND, IDM_CLOSEALL, 0L );
    if( GetWindow( hwndClient, GW_CHILD ) != NULL )
        return 0;
        break;

    case WM_DESTROY:
        PostQuitMessage( 0 );
        return 0;

    }

return DefFrameProc( hwnd, hwndClient, message, wParam, lParam );
}

//-------------------------------------------------
//--- CloseEnumProc -- Пытается разрушить дочернее окно
BOOL FAR PASCAL CloseEnumProc( HWND hwnd, LONG lParam )
{

if( GetWindow( hwnd, GW_OWNER ))
    return 1;

SendMessage( GetParent( hwnd ), WM_MDIRESTORE, hwnd, 0L );

if( !SendMessage( hwnd, WM_QUERYENDSESSION, 0, 0L ))
    return 1;

SendMessage( GetParent( hwnd ), WM_MDIDESTROY, hwnd, 0L );
    return 1;

}
```

Листинг 3. Оконная процедура для класса MDITextChild

```
//-------------------------------------------------
//--- TextWndProc - Оконная процедура для класса MDITextChild
long FAR PASCAL TextWndProc(    HWND hwnd, WORD message, WORD wParam,
                                LONG lParam )
{

static WORD nFontFamily[FAMILYNUM]    = { FF_ROMAN,
                                          FF_SWISS,
                                          FF_MODERN,
                                          FF_SCRIPT,
                                          FF_DECORATIVE };
static char *szFaceName[]             = { "Times Roman",
                                          "Helvetica",
                                          "Courier",
                                          "Cursive",
                                         "Old English" };
static int nCharSet[FAMILYNUM]        = { ANSI_CHARSET,
                                          ANSI_CHARSET,
                                          ANSI_CHARSET,
                                          OEM_CHARSET,
                                          ANSI_CHARSET };

static HWND hwndFrame, hwndClient;
HDC hdc;
HMENU hMenu;
LOCALHANDLE hTextData;
NPTEXTDATA npTextData;
PAINTSTRUCT ps;
RECT rect;
HFONT hFont;
char szBuff[] = "                              ";

switch( message ) {

    case WM_CREATE:
        hTextData = LocalAlloc(LMEM_MOVEABLE | LMEM_ZEROINIT,
                               sizeof( TEXTDATA ));
        npTextData = (NPTEXTDATA)LocalLock( hTextData );
        npTextData->nItem = IDM_ROMAN;
        npTextData->logFnt.lfHeight         = 20;
        npTextData->logFnt.lfWidth          = 16;
        npTextData->logFnt.lfEscapement     = 0;
        npTextData->logFnt.lfOrientation    = 0;
        npTextData->logFnt.lfWeight     = FW_NORMAL;
        npTextData->logFnt.lfItalic     = 0;
        npTextData->logFnt.lfUnderline  = 0;
        npTextData->logFnt.lfStrikeOut  = 0;
        npTextData->logFnt.lfCharSet    = nCharSet[0];
        npTextData->logFnt.lfOutPrecision = OUT_DEFAULT_PRECIS;
        npTextData->logFnt.lfClipPrecision = CLIP_DEFAULT_PRECIS;
        npTextData->logFnt.lfQuality    = DEFAULT_QUALITY;
        npTextData->logFnt.lfPitchAndFamily = nFontFamily[0];
        strcpy( npTextData->logFnt.lfFaceName, szFaceName[0] );

        LocalUnlock( hTextData );

        SetWindowWord( hwnd, 0, hTextData );

        hwndClient = GetParent( hwnd );
        hwndFrame = GetParent( hwndClient );

        return 0;

case WM_COMMAND:
    switch( wParam ) {

        case IDM_ROMAN:
        case IDM_SWISS:
        case IDM_MODERN:
        case IDM_SCRIPT:
        case IDM_DECORATIVE:
            hTextData = GetWindowWord( hwnd, 0 );
            npTextData = (NPTEXTDATA)LocalLock( hTextData );

            hMenu = GetMenu( hwndFrame );

            CheckMenuItem(  hMenu,
                            npTextData->nItem,
                            MF_UNCHECKED );
            npTextData->nItem = wParam;
            CheckMenuItem(  hMenu,
                            npTextData->nItem,
                            MF_CHECKED );

        npTextData->logFnt.lfCharSet = nCharSet[wParam-IDM_ROMAN];
        npTextData->logFnt.lfPitchAndFamily =   nFontFamily[wParam-
                                                IDM_ROMAN];
            strcpy(  npTextData->>logFnt.lfFaceName,
                     szFaceName[wParam-IDM_ROMAN] );

            LocalUnlock( hTextData );
            InvalidateRect( hwnd, NULL, FALSE );

        }
        return 0;

    case WM_PAINT:
        hdc = BeginPaint( hwnd, &ps );

        hTextData = GetWindowWord( hwnd, 0 );
        npTextData = (NPTEXTDATA)LocalLock( hTextData );
        LocalUnlock( hTextData );

        hFont = SelectObject( hdc,
                CreateFontIndirect( &(npTextData->logFnt) ));

        GetClientRect( hwnd, &rect );
        DrawText(   hdc,
                    szBuff,
                    -1,
                    &rect,
                    DT_SINGLELINE | DT_CENTER | DT_VCENTER );

        DrawText(   hdc,
                    npTextData->logFnt.lfFaceName,
                    -1,
                    &rect,
                    DT_SINGLELINE | DT_CENTER | DT_VCENTER );

        DeleteObject( SelectObject( hdc, hFont ));
        EndPaint( hwnd, &ps );

        return 0;

    case WM_MDIACTIVATE:

        if( wParam == TRUE )
            SendMessage( hwndClient, WM_MDISETMENU, 0,
                         MAKELONG( hMenuText, hMenuTextWindow ));

        hTextData = GetWindowWord( hwnd, 0 );
        npTextData = (NPTEXTDATA)LocalLock( hTextData );
        CheckMenuItem(  hMenuText, npTextData->nItem,
                        wParam ? MF_CHECKED : MF_UNCHECKED );
        LocalUnlock( hTextData );
```

```
            if( wParam = = FALSE )
                SendMessage( hwndClient, WM_MDISETMENU, 0,
                             MAKELONG( hMenuInit, hMenuInitWindow ));

            DrawMenuBar( hwndFrame );
            return 0;

        case WM_QUERYENDSESSION:
        case WM_CLOSE:
            if(  MessageBox(  hwnd, "Ok to close window ?", "Text Style",
                              MB_ICONQUESTION | MB_OKCANCEL )
                 != IDOK )
                return 0;
            break;

        case WM_DESTROY:
            hTextData = GetWindowWord( hwnd, 0 );
            LocalFree( hTextData );
            return 0;

        }

    return DefMDIChildProc( hwnd, message, wParam, lParam );
    }
```

Листинг 4. Оконная процедура для класса MDIEmptyChild

```
//-------------------------------------------------
//--- EmptyWndProc - Оконная процедура для класса MDIEmptyChild
long FAR PASCAL EmptyWndProc(   HWND hwnd, WORD message, WORD wParam,
                                LONG lParam )
{

static HWND hwndFrame, hwndClient;

switch( message ) {

    case WM_CREATE:
        hwndClient = GetParent( hwnd );
        hwndFrame = GetParent( hwndClient );

        return 0;

    case WM_MDIACTIVATE:
        if( wParam = = TRUE )
        SendMessage( hwndClient, WM_MDISETMENU, 0,
                     MAKELONG( hMenuEmpty, hMenuEmptyWindow ));
    else
        SendMessage( hwndClient, WM_MDISETMENU, 0,
                     MAKELONG( hMenuInit, hMenuInitWindow ));

    DrawMenuBar( hwndFrame );
    return 0;

    }

return DefMDIChildProc( hwnd, message, wParam, lParam );
}
```

Листинг 5. Файл заголовка SMDI.H

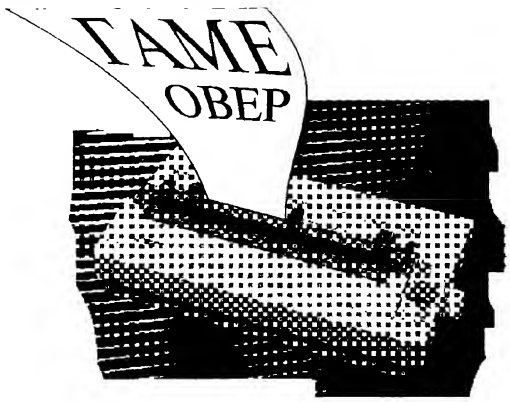
```
//-------------------------------------------------
//--- SMDI.H - файл заголовка для SMDI.C
//--- Copyright (c) D.Plastinin 1992

//---Структура для сохранения уникальной информации о шрифте
typedef struct {
    int nItem;
    LOGFONT logFnt;
} TEXTDATA;

typedef TEXTDATA NEAR *NPTEXTDATA;

//---Позиции пункта "WINDOW" в меню главного окна
#define  INIT_MENU_POS 0
#define  TEXT_MENU_POS 2
#define  EMPTY_MENU_POS 1

//---Инициализаторы пунктов меню
#define  IDM_NEWTEXT        10
#define  IDM_NEWEMPTY       11
#define  IDM_CLOSE          12
#define  IDM_EXIT           13

#define  IDM_ROMAN          20
#define  IDM_SWISS          21
#define  IDM_MODERN         22
#define  IDM_SCRIPT         23
#define  IDM_DECORATIVE     24

#define  IDM_TILE           30
#define  IDM_CASCADE        31
#define  IDM_ARRANGE        32
#define  IDM_CLOSEALL       33

//---Идентификатор первого дочернего окна
#define  IDM_FIRSTCHILD     100

//---Количество гарнитур текстов
#define  FAMILYNUM          5

//---Прототипы функций
long     FAR PASCAL  FrameWndProc( HWND, WORD, WORD, LONG );
BOOL     FAR PASCAL  CloseEnumProc( HWND, LONG );
long     FAR PASCAL  TextWndProc( HWND, WORD, WORD, LONG );
long     FAR PASCAL  EmptyWndProc( HWND, WORD, WORD, LONG );

//---Глобальные переменные
HANDLE   hInst;
HMENU    hMenuInit, hMenuText, hMenuEmpty;
HMENU    hMenuInitWindow, hMenuTextWindow, hMenuEmptyWindow;
```

Листинг 6. Файл ресурсов SMDI.RS

```
MDIMENUEMPTY MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "New &Text", 10
        MENUITEM "New &Empty", 11
        MENUITEM "&Close", 12
        MENUITEM SEPARATOR
        MENUITEM "E&xit", 13
    END

    POPUP "&Window"
    BEGIN
        MENUITEM "&Cascade\tShift+F5", 31
        MENUITEM "&Tile\tShift+F4", 30
        MENUITEM "Arrange &Icons", 32
        MENUITEM "Close &All", 33
    END
END

MDIMENUINIT MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "New &Text", 10
        MENUITEM "New &Empty", 11
        MENUITEM SEPARATOR
        MENUITEM "E&xit", 13
    END
END

MDIMENUTEXT MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "New &Text", 10
        MENUITEM "New &Empty", 11
        MENUITEM "&Close", 12
        MENUITEM SEPARATOR
        MENUITEM "E&xit", 13
    END

    POPUP "&Text"
    BEGIN
        MENUITEM "&Roman", 20
        MENUITEM "&Swiss", 21
        MENUITEM "&Modern", 22
        MENUITEM "S&cript", 23
        MENUITEM "&Decorative", 24
    END

    POPUP "&Window"
    BEGIN
        MENUITEM "&Cascade\tShift+F5", 31
        MENUITEM "&Tile\tShift+F4", 30
        MENUITEM "Arrange &Icons", 32
        MENUITEM "Close &All", 33
    END
END

MDIACCEL ACCELERATORS
BEGIN
    VK_F5, 31, VIRTKEY, SHIFT
    VK_F4, 30, VIRTKEY, SHIFT
END
```

Листинг 7. Файл определения модуля SMDI.DEF

```
NAME            SMDI
DESCRIPTION     "Multi Document Interface"
EXETYPE         WINDOWS
STUB            'WINSTUB.EXE'
CODE            PRELOAD MOVEABLE DISCARDABLE
DATA            PRELOAD MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE       8192
EXPORTS         FRAMEWNDPROC
                TEXTWNDPROC
                CLOSEENUMPROC
```



Программная русификация матричных принтеров

О. Шарапов

Быстрое распространение в нашей стране зарубежных компьютеров IBM PC, XT, AT и совместимых с ними, кроме всем известных положительных эффектов, заставило нас столкнуться и с некоторыми трудностями, связанными с их эксплуатацией.

Одна из них связана с тем, что рынок производителей компьютерной техники не был вовремя сориентирован на ее сбыт в странах Восточной Европы и, в частности, славяноязычных странах. Поэтому периферийное оборудование компьютеров (клавиатура, дисплей, принтер и тому подобное) не позволяет без специальной аппаратной поддержки или программного обеспечения оперировать символами кириллицы. Наиболее уязвимыми в этом отношении оказались принтеры — в очереди за адаптацией их к отечественным условиям они занимали одно из последних мест.

В настоящее время эта проблема уже частично решена. Фирмы-изготовители компьютерной техники стали выпускать принтеры, знакогенераторы которых включают в себя символы кириллицы, да и программисты написали соответствующее программное обеспечение. Последнее может быть как универсальным (пакеты BETA, LETTRIX, LEXICON и так далее), имеющим возможность настраиваться на различные типы принтеров, или узконаправленным (например, пакет EPSONFX), предназначенным для работы с какой-то одной маркой принтера.

Но даже существование определенного программного обеспечения, позволяющего распечатывать кириллицу на принтерах, проблему их русификации еще не сняло полностью, потому что данные программные средства зачастую имеют те или иные погрешности или не подходят для какого-то конкретного случая по своим параметрам (скорость вывода символов на печать; ограничения, накладываемые при работе в фоновом режиме; объем памяти, занимаемый резидентной частью программы и тому подобное). Так что, приобретя компьютер с нерусифицированным принтером, Вы просто можете оказаться в очень затруднительном положении. Программистам, да и пользователям персональных компьютеров можно посоветовать в этом случае на выбор один из трех способов решения этой проблемы:

1) замену ПЗУ (постоянного запоминающего устройства) знакогенератора принтера на новое, с прошитыми в нем символами кириллицы;
2) установку блоков расширения памяти-картриджей в принтерах, имеющих аппаратные возможности для этого;
3) использование с учетом ранее изложенных замечаний уже существующей или же написание собственной специальной программы, позволяющей распечатывать русские тексты.

Вкратце рассмотрим достоинства и недостатки этих способов. Итак, первый способ. Он потребует определенных денежных вложений и поиска фирмы, обладающей микросхемой ПЗУ именно той марки, которая использована в вашем принтере. Но при применении этого способа может возникнуть такая ситуация, что, установив новое ПЗУ с прошитой кириллицей на месте специфических германских символов, Вам необходимо будет распечатать немецкий текст. Вы опять попадете в трудноразрешимую ситуацию.

Такого не произойдет при использовании Вами второго из перечисленных способов. Тогда после установки картриджа с записанным в нем шрифтом кириллицы, манипулируя набором определенных DIP переключателей принтера (или используя специфические ESC последовательности в качестве псевдопереключателей), можно активизировать тот или иной доступный набор символов. Этот способ хорош, но ведет к более крупным денежным затратам, чем при использовании первого способа, да и к тому же не у каждого принтера имеются аппаратные возможности для установки картриджа.

Поэтому хотелось бы сориентировать Вас на использование третьего из предложенных способов.

Для тех, кто решил написать программу русификации своего принтера самостоятельно, предназначена эта статья.

Перед составлением алгоритма и собственно написанием программы-драйвера русской печати необходимо внимательно изучить инструкцию на приобретенный принтер. В процессе изучения определите, к какой из нижеперечисленных категорий он относится.

Для этого необходимо провести небольшой экскурс во внутреннее устройство принтера, знание которого будет необходимо при создании программ-русификаторов.

Принтеры имеют в своем распоряжении ПЗУ, в котором, кроме необходимой для работы принтера, но не интересующей нас информации, хранятся также байты конфигурации стандартных символов (набор которых приводится в одном из приложений инструкции по эксплуатации Вашего принтера). В его распоряжении имеется и оперативное запоминающее устройство (ОЗУ), предназначенное, в частности, для хранения байтов конфигурации символов кириллицы, которые Вам еще предстоит определить. Важнейшей характеристикой ОЗУ является его объем, измеряемый количеством символов, которые могут быть в него загружены. Именно по этой характеристике — объем ОЗУ — классифицировались принтеры.

Классификация принтеров по способу их русификации

I. Принтеры, имеющие ОЗУ.
   1.1 Имеется ОЗУ, предназначенное для хранения конфигурации всейASCII таблицы (256 символов).
   1.2 Имеется ОЗУ, предназначенное для хранения конфигурации более 64 символов.
   1.3 Имеется ОЗУ, предназначенное для хранения конфигурации более 40 символов.
   1.4 Имеется ОЗУ, предназначенное для хранения конфигурации менее 40 символов.
II. Принтеры, не имеющие ОЗУ.

Русификация принтеров, имеющих объем ОЗУ, предназначенный для хранения полной ASCII таблицы, наиболее проста. К сожалению, принтеров, относящихся к данной группе, не так много (к ней относится, например, принтер EPSON NX-15). Для его русификации может быть предложен следующий алгоритм (блок-схема которого показана на рис. 1):

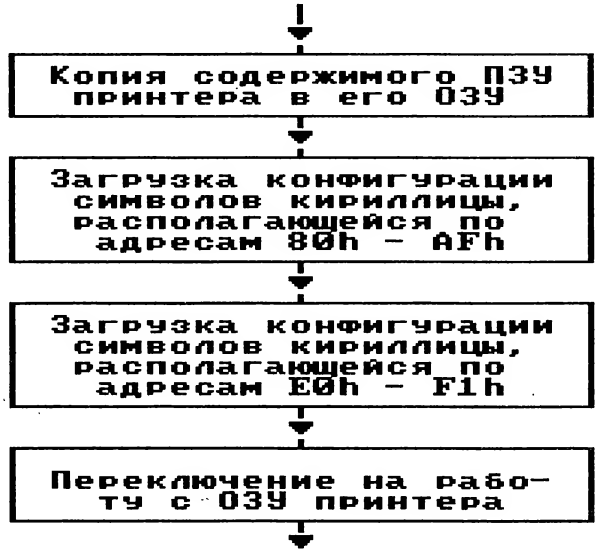


Рис. 1. Блок-схема алгоритма русификации принтера EPSON NX-15

- копирование содержимого ПЗУ принтера в его ОЗУ;
- заполнение ячеек памяти ОЗУ принтера (предназначенных для хранения символов с кодами 80h-AFh и E0h-F1h) байтами конфигурации определяемых Вами символов;
- переключение принтера на работу с его ОЗУ.

В результате выполнения действий, описанных в пункте 2, в ОЗУ принтера будет располагаться ASCII таблица, в которой на месте специфических германских символов будут размещены символы кириллицы (табл. 1).

Все изложенные действия производятся с помощью пересылки определенных управляющих последовательностей (ESC последовательностей) на принтер. Реализация такого алгоритма, с точки зрения программирования, не представляет особых сложностей, поэтому листинг программы опускается.

Таблица 1

Размещение символов кириллицы в ASCII таблице

Десятичное значение		128	144	160	176	192	208	224	240
	Шестнадцатеричное значение	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
0	x0	А	Р	а				р	
1	x1	Б	С	б				с	
2	x2	В	Т	в				т	
3	x3	Г	У	г				у	
4	x4	Д	Ф	д				ф	
5	x5	Е	Х	е				х	
6	x6	Ж	Ц	ж				ц	
7	x7	З	Ч	з				ч	
8	x8	И	Ш	и				ш	
9	x9	Й	Щ	й				щ	
10	xA	К	Ъ	к				ъ	
11	xB	Л	Ы	л				ы	
12	xC	М	Ь	м				ь	
13	xD	Н	Э	н				э	
14	xE	О	Ю	о				ю	
15	xF	П	Я	п				я	

Примечание. Будьте внимательны, определенный DIP переключатель в Вашем принтере отвечает за возможность загрузки в ОЗУ принтера конфигурации загружаемых символов. Перед запуском программы-русификатора убедитесь в том, что загрузка кириллицы в ОЗУ аппаратно разрешена. Это касается всех принтеров, русификация которых производится при использовании их ОЗУ.

Несколько сложнее написание программы-русификатора для принтера, имеющего объем ОЗУ, вмещающий конфигурацию менее 256 символов. Алгоритм функционирования драйвера для такого принтера принципиально отличается от изложенного выше.

Часть из них была выделена в самостоятельную группу, у принтеров которой "...имеется ОЗУ, предназначенное для хранения конфигурации более 64 символов". Почему именно 64 символа? Потому что в русском алфавите, без учета буквы 'ё', которая часто игнорируется как символ и заменяется буквой 'е', ровно 32 буквы, которые в сочетании "заглавная + прописная" дадут 64 символа. То есть для создания и хранения конфигурации символов русского алфавита в указанном сочетании необходим объем ОЗУ принтера, предназначенный для хранения не менее 64 символов. В чем же заключается главное отличие русификации данного типа принтера (к нему относятся, например, принтеры марок EPSON LQ-500, STAR LCQ-10, STAR NX-1500 и другие) от способа, описанного выше?

Это отличие заключается в том, что в процессе вывода текстов на печать у пользователя отсутствует возможность работать лишь только с ОЗУ принтера из-за ограниченности его объема. В него могут быть помещены символы кириллицы, какие-нибудь специфические определяемые пользователем знаки, но нет возможности переписать в ОЗУ принтера конфигурацию всех символов, хранящихся в его ПЗУ. Именно это делает бесполезным копирование содержимого ПЗУ принтера в его ОЗУ, чем пользовались при русификации принтеров, имеющих ОЗУ, предназначенное для хранения конфигурации 256 символов.

Для распечатывания текстов на рассматриваемом типе принтеров должны происходить: опережающий анализ символа, выводимого на печать, и в зависимости от результата анализа ("кириллица" — "не кириллица") — автоматическое переключение режима функционирования принтера: либо на работу с ПЗУ, либо на работу с ОЗУ.

Этот анализ должен производиться автоматически при выводе на печать очередного символа. А так как распечатывание текстов (при помощи системных команд COPY или TYPE или с помощью файла PRINT.COM) осуществляется функцией 0h прерывания INT17h, необходимо заменить стандартную нулевую функцию этого прерывания на свою собственную, написанную с учетом требований, предъявляемых к ней.

Поэтому программа-русификатор принтера состоит из двух частей: установочной и рабочей. Первая из них осуществляет следующие действия (блок-схема алгоритма установочной части программы представлена на рис. 2):

- определяет адрес входа в стандартный обработчик прерывания INT17h и запоминает его в определенных ячейках памяти (для этого резервируются два слова — rem_17_off и rem_17_seg) рабочей части программы;
- переустанавливает адрес входа в прерывание INT17h на адрес входа в процедуру new_int17 (модернизированный для нашей задачи стандартный обработчик прерывания INT17h);
- загружает пользовательский шрифт в ОЗУ принтера;
- оставляет рабочую часть программы резидентной.

Рис. 2. Блок-схема алгоритма рабочей части программы-русификатора принтера STAR LC-10

Рабочая часть программы, остающаяся резидентно размещенной в памяти компьютера, берет на себя основную функцию русификации указанного выше типа принтеров, а именно:

1) определение номера функции вызываемого прерывания и в случае ненулевой функции выполнение пункта 5;
2) разделение выводимых на печать символов по признаку — "кириллица" — "не кириллица";
3) перекодировку в случае символа кириллицы;
4) переключение на работу с ОЗУ или с ПЗУ принтера на основе результатов сортировки;
5) переход на стандартный обработчик прерывания INT 17h.

Под перекодировкой понимается некоторая процедура согласования между символами, которыми набран текст, выводимый на печать, и символами, которыми этот текст будет отображен на распечатке.

Исходной информацией при выводе символа на печать является его ASCII код — уникальный признак, однозначно определяющий символ. Таким образом, зная ASCII коды символов кириллицы и номера ячеек (буферов) ОЗУ принтера, в которых хранятся байты данных, определяющих конфигурацию соответствующих символов кириллицы, можно написать процедуру перекодировки ASCII кодов символов кириллицы в номера соответствующих ячеек ОЗУ принтера, в которых хранится конфигурация этих символов.

Так как в принтере STAR LC-10 существует возможность загрузки конфигурации определяемых пользователем символов с 32-й по 127-ю ячейки ОЗУ

включительно[4], принимаем, что конфигурация символов кириллицы загружается в него, начиная с 62-ой ячейки, в следующем порядке (табл. 2).

Таблица 2

Размещение конфигурации символов кириллицы в ячейках ОЗУ принтера STAR LC-10

Десятичное значение	60	70	80	90	100	110	120
0		И	Т	Ь	ж	р	ъ
1		Й	У	Э	з	с	ы
2	А	К	Ф	Ю	и	т	ь
3	Б	Л	Х	Я	й	у	э
4	В	М	Ц	а	к	ф	ю
5	Г	Н	Ч	б	л	х	я
6	Д	О	Ш	в	м	ц	
7	Е	П	Щ	г	н	ч	
8	Ж	Р	Ъ	д	о	ш	
9	З	С	Ы	е	п	щ	

Теперь необходимо поставить в соответствие друг другу ASCII коды символов кириллицы, поступающих во входной порт принтера, и номера ячеек ОЗУ принтера, в которых хранится конфигурация соответствующих русских символов. Внимательное изучение расположения кириллицы в табл. 1 и табл. 2 дает номер ячейки ОЗУ принтера, в которой расположены байты данных конфигурации выводимого на печать символа "кириллицы", определяющийся как:

1) ASCII код символа минус 66 (для символов от 'А' до 'п');

2) ASCII код символа минус 114 (для символов от 'р' до 'я').

Блок-схема алгоритма пользовательского обработчика прерывания INT17h, реализующего предложенные идеи, представлена на рис. 3. Программа-русификатор принтера STAR LC-10 написана на ассемблере (листинг 1) и сопровождена необходимыми для ее читабельности комментариями.

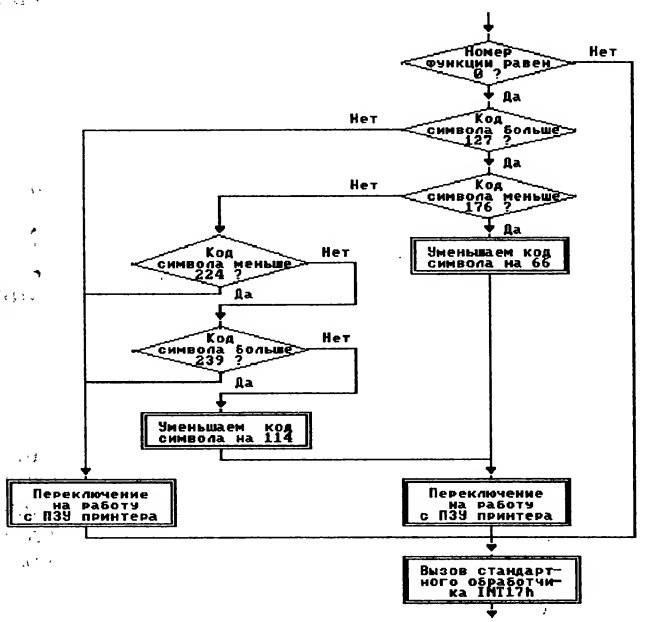


Рис. 3. Блок–схема алгоритма пользовательского обработчика прерывания INT17h (STAR LC–10)

Еще сложнее алгоритм, применяемый для русификации принтеров, объем ОЗУ которых предназначен для хранения конфигурации от 40 до 63 символов кириллицы включительно. Прежде всего необходимо объяснить, почему принтеры с таким объемом были выделены в самостоятельную группу при их классификации.

Максимальное число символов — 63 ограничивается, исходя из того соображения, что для принтеров, имеющих ОЗУ большей емкости, применим алгоритм русификации, описанный ранее, более короткий и простой.

Минимальное число символов — 40 определяется в ходе приведенных ниже рассуждений.

В основу русификации принтеров, имеющих ограниченное ОЗУ, положен принцип аналогов [2], основанный на подмене некоторых символов кириллицы подобными им по конфигурации латинскими символами. Возможные варианты подмен отражены в табл. 3.

Таким образом, 22 символа кириллицы могут быть продублированы символами-аналогами из левой половины ASCII таблицы. С учетом изложенного выше и того, что допускается замена символа "Й" на символ "И", у нас остаются 40 символов кириллицы, конфигурацию которых мы должны поместить в ОЗУ принтера.

Русификация данного типа принтеров также состоит из двух частей: установочной и рабочей.

Концепция установочной части полностью соответствует блок-схеме, приведенной на рис. 2.

Рабочая часть, представляющая собой модернизированный в соответствии с требованиями пользовательский обработчик прерывания INT 17h, после завершения инсталляционной программы остается в памяти (до перезагрузки системы, сбоя по питанию, некорректного исполнения каких-нибудь пользовательских программ и тому подобного) и берет на себя следующие обязанности:

1) разделение символьного потока, выводимого на печать, по признаку "кириллица" — "не кириллица". В случае, если символ — "не кириллица", переключение на работу с ПЗУ принтера. Затем выполнение пункта 5;

2) если символ — "кириллица", производим его идентификацию на предмет наличия у него аналога, согласно табл. 3. В случае, если имеется аналог, производятся последовательно следующие действия:

– замена кода кириллицы на код символа-аналога;

– переключение на работу с ПЗУ принтера;

– выполнение пункта 5;

3) если аналог отсутствует, производится замена кода кириллицы на номер ячейки ОЗУ принтера, в которой располагаются байты, определяющие конфигурацию этого символа;

4) переключение на работу с ОЗУ принтера;

5) переход к стандартному обработчику прерывания INT 17h.

Необходимо отметить, что изложенный алгоритм русификации применим и для принтеров, имеющих объем ОЗУ, предназначенный для хранения конфигурации более 63 символов. В этом случае необходимо только проследить, чтобы верхняя и нижняя границы начертания символов-аналогов и символов кириллицы, определяемых Вами, совпадали.

Таблица 3

Свод символов-аналогов

Символ кириллицы	Код символа-аналога	Символ кириллицы	Код символа-аналога	Символ кириллицы	Код символа-аналога
А	41h	Ц		м	
Б		Ч		н	
В	42h	Ш		о	6Fh
Г		Щ		п	6Eh
Д		Ъ		р	70h
Е	45h	Ы		с	63h
Ж		Ь		т	6Dh
З	33h	Э		у	79h
И, й		Ю		ф	
К	4Bh	Я		х	78h
Л		а	61h	ц	
М	4Dh	б		ч	
Н	48h	в		ш	
О	4Fh	г		щ	
П		д		ъ	
Р	50h	е	65h	ы	
С	43h	ж		ь	
Т	54h	з		э	
У		и, й	75h	ю	
Ф		к		я	
Х	48h	л			

Для наглядности рассмотрим программу русификации изложенным методом принтера EPSON LQ-500. Изначально определяем, что байты конфигурации символов кириллицы, не имеющие аналогов в левой половине ASCII таблицы, расположены, начиная с 62-й ячейки ОЗУ принтера [3].

Определение наличия аналога у символа кириллицы производится с помощью выборки информационного байта из таблицы Decod со смещением, равным преобразованному коду кириллицы, от ее начала. Для правильного функционирования программы таблица Decod составляется следующим образом:

- длина таблицы — 64 байта, символу "А" (80h) в ней соответствует нулевой байт, символу "я" (0EFh) соответствует 63-й байт;
- заполнение таблицы производится следующим образом: для кириллицы, имеющей символ-аналог, в таблице Decod по соответствующему смещению записывается ASCII код последнего; в противном случае в таблицу Decod записывается 0.

Поэтому программное определение наличия аналога у кириллицы производится обыкновенным тестированием байта из таблицы Decod со смещением, равным преобразованному коду кириллицы от ее начала, на ноль. Если считанный байт равен 0 — аналога у кириллицы нет.

Преобразованный код кириллицы получается с помощью уменьшения кода символа на:

- 128 (для символов, лежащих в пределах 80h — 0AFh);
- 176 (для символов, лежащих в пределах 0E0h — 0EFh).

В случае если кириллица не имеет аналога, из таблицы TabRam по смещению, равному преобразованному коду кириллицы, извлекается байт, содержащий номер ячейки ОЗУ принтера, в которой хранится конфигурация этого символа.

Таблица TabRam составляется следующим образом:

- байты, соответствующие символам кириллицы, имеющим аналоги, заполняются нулями;
- в противном случае байту присваивается номер ячейки ОЗУ, в которой хранятся байты конфигурации кириллицы.

Табл. 4 наглядно демонстрирует правильное составление байтовых последовательностей Decod и TabRam.

Программа на ассемблере пользовательского обработчика INT 17h, а также шрифты символов кириллицы, не имеющих аналогов, для принтера EPSON LQ-500 представлена в листинге 2.

К последней группе матричных печатающих устройств, в соответствии с их классификацией, относятся принтеры, которые не обладают ОЗУ вообще, либо принтеры, объем ОЗУ которых предназначен для размещения в нем менее 40 байтов конфигурации определяемых пользователем символов. Откровенно говоря, принтеры без ОЗУ мне встречать не приходилось, но допустить такую возможность вполне логично. Что касается принтеров с малым объемом оперативного запоминающего устройства, то к ним могут быть отнесены устройства, представленные в приведенной далее табл. 5. В колонке "размер ОЗУ" обозначена величина, определяющая максимальное число нестандартных символов, конфигурация которых может быть задана пользователем.

Что же делать, если ни один из ранее описанных способов, обеспечивающих вывод кириллицы на принтер с использованием его ОЗУ, не подходит (исключение составляет специально разработанный вероятностный метод вывода русских символов на принтерах, имеющих объем ОЗУ, позволяющий хранить конфигурацию менее 40 символов; с помощью этого метода, например, был русифицирован принтер STAR NB24 — 10/15)? В этом случае к Вам на помощь придет графика, а именно возможность принтера распечатывать различные предварительно закодированные графические изображения. Графические изображения, распечатывание которых возможно на принтере, в более детальном рассмотрении можно разбить на две группы:

- графические изображения в самом прямом смысле слова. Это рисунки, графические, коммерческие диаграммы, копии фотографий, созданные сканерами, нестандартные шрифты (например, графические шрифты пакетов Turbo C, Turbo Pascal) и тому подобное;
- унифицированные графические изображения. К ним можно отнести шрифты, с помощью которых предлагается производить русификацию принтеров рассматриваемого типа. Все символы такого шрифта можно рассматривать как некоторые изображения, которые были выделены в отдельную группу по следующим причинам:
  а) частый вывод каждого из них на печать;
  б) стандартные размеры матрицы, в рамках которой определяется конфигурация символа (справедливо только для фиксированных шрифтов), а следовательно и одинаковое число байтов, определяющих конфигурацию каждого из них;

Таблица 4

Наборы байтовых последовательностей таблиц Decod и TabRam

Смещение в таблицах	Символ кириллицы	Decod	TabRam	Смещение в таблицах	Символ кириллицы	Decod	TabRam
0	А	41h	0	32	а	61h	0
1	Б	0	62	33	б	0	81
2	В	42h	0	34	в	0	82
3	Г	0	63	35	г	0	83
4	Д	0	64	36	д	0	84
5	Е	45h	0	37	е	65h	0
6	Ж	0	65	38	ж	0	85
7	З	33h	0	39	з	0	86
8	И	0	66	40	и	75h	0
9	Й	0	66	41	й	75h	0
10	К	4Bh	0	42	к	0	87
11	Л	0	67	43	л	0	88
12	М	4Dh	0	44	м	0	89
13	Н	48h	0	45	н	0	90
14	О	4Fh	0	46	о	6Fh	0
15	П	0	68	47	п	6Eh	0
16	Р	50h	0	48	р	70h	0
17	С	43h	0	49	с	63h	0
18	Т	54h	0	50	т	6Dhh	0
19	У	0	69	51	у	79h	0
20	Ф	0	70	52	ф	0	91
21	Х	58h	0	53	х	78h	0
22	Ц	0	71	54	ц	0	92
23	Ч	0	72	55	ч	0	93
24	Ш	0	73	56	ш	0	94
25	Щ	0	74	57	щ	0	95
26	Ъ	0	75	58	ъ	0	96
27	Ы	0	76	59	ы	0	97
28	Ь	0	77	60	ь	0	98
29	Э	0	78	61	э	0	99
30	Ю	0	79	62	ю	0	100
31	Я	0	80	63	я	0	101

в) возможность предопределения конфигурации изображения заранее.

Ярким примером реализации возможностей принтера при распечатывании изображений первой группы является создание твердой графической копии экрана, осуществляемое с помощью резидентной программы-утилиты GRAPHICS.COM или некоторой иной программы. Алгоритм создания таких изображений значительно сложнее рассматриваемого в этой статье, базируется на совершенно иной концепции формирования приемлемой для принтера кодировки выводимого на печать изображения. Поэтому рассмотрение процесса создания копии графического изображения на принтере в этой статье опускается.

Таблица 5

Принтеры с малым объемом ОЗУ

Наименование принтера	Размер ОЗУ
STAR NB24 — 10/15	35
STAR LQ-10	6
STAR SP312F	10

Вывод же на принтер графических изображений второй группы превосходно осуществляет широко известный пакет LETTRIX. При автоматизированном с помощью этого пакета создании шрифтов кириллицы производится настройка на тип принтера, где будет осуществляться распечатка русских текстов. Хотелось бы отметить, что при правильном использовании этого пакета кириллица будет выводиться на печать практически на любом принтере.

Тогда (вполне логично) Вы можете задать вопрос: "Зачем же было огород городить: создавать, хоть и не очень сложные, алгоритмы, ориентирующиеся на использование ОЗУ принтера, если графические возможности последнего позволяют делать то же самое и меньшей кровью?"

Ответ прост. Функционирование принтера в графическом режиме имеет несколько недостатков, основные из них следующие:

– скорость вывода графических символов на печать намного ниже, чем при применении пользовательского шрифта, загруженного в ОЗУ, а тем более шрифта, прошитого в ПЗУ принтера;

Листинг 1. Программа русификации принтера STAR LC–10

```
title star_lc.asm
command segment
        assume  cs:command,ds:command
;
        jmp Begin
rem_17_off  dw  ?           ; смещение стандартного обработчика
rem_17_seg  dw  ?           ; сегмент стандартного обработчика
;
;= = = = = = = = = = Секция обработки INT 17h = = = = = = = = = = = =
;
new_int17   proc far
            cmp ah,0h       ; функция 0 прерывания INT17h?
            jnz work1       ; нет, переход на *work1
            cmp al,127      ; код символа больше 127
            jng en          ; нет - переход на *en
            cmp al,176      ; код символа меньше 176
            jb  trans       ; да - переход на *trans
            cmp al,224      ; код символа меньше 224
            jb  en          ; да - переход на *en
            cmp al,239      ; нет, код символа больше 239
            jg  en          ; да - переход на *en
            sub al,114      ; уменьшить код символа на 114
            jmp work2       ; переход на *work2
trans:      sub al,66       ; уменьшить код символа на 66
            jmp work2       ; переход на  *work2
en:         push ax
            call rom        ; переключение на работу с ПЗУ
            jmp nowork1
work2:      push ax
            call ram        ; переключение на работу с ОЗУ
nowork1:    pop ax
work1:      mov dx,0h
            jmp dword ptr rem_17_off   ; вызов старого вектора INT17h
new_int17   endp
;
ram         proc near       ; процедура переключения на
            mov al,1bh      ; работу с ОЗУ
            call print
            mov al,25h
            call print
            mov al,31h
            call print
            ret
ram         endp
;
rom         proc near       ; процедура переключения на
            mov al,1bh      ; работу с ПЗУ
            call print
            mov al,25h
            call print
            mov al,30h
            call print
            ret
rom         endp
;
print       proc near       ; процедура вывода символа
            mov dx,378h     ; на принтер
            out dx,al
            inc dx
waitp:      in  al,dx
            test al,80h
            jz  waitp
            inc dx
            mov al,0dh
            out dx,al
            mov al,0ch
            out dx,al
            ret
print       endp
;
load        proc near       ; процедура загрузки черновой
            xor dx,dx       ; конфигурации кириллицы
met:        push dx
            lea si,cs:cod
            inc byte ptr cs:[si+3]
            inc byte ptr cs:[si+4]
            mov cx,05h
in0:        mov al,byte ptr cs:[si]
            call print
            inc si
            loop in0
            pop dx
            push dx
            lea si,cs:cod
            mov ax,12
            mul dx
            add si,5
            add si,ax
            mov cl,12
in1:        mov al,byte ptr cs:[si]
            call print
            inc si
            loop in1
            pop dx
            inc dx
            cmp dx,64
            jnz met
            ret
load        endp
cod         db  1bh,26h,00h,61,61
;STAR-LC 10
db  8ah,00h,06h,18h,60h,88h,00h,88h,00h,0feh,00h,00h    ; символ А *
db  8ah,00h,0feh,00h,92h,00h,92h,00h,92h,0ch,00h,00h    ; символ Б *
db  8ah,00h,0feh,00h,92h,00h,92h,00h,92h,6ch,00h,00h    ; символ В *
db  8ah,00h,0feh,00h,80h,00h,80h,00h,80h,40h,00h,00h    ; символ Г *
db  8ah,00h,03h,3ch,42h,00h,82h,00h,0fch,03h,00h,00h    ; символ Д *
db  8ah,00h,0feh,00h,92h,00h,92h,00h,82h,44h,00h,00h    ; символ Е *
db  8ah,00h,00h,0c6h,28h,10h,0eeh,10h,28h,0c6h,00h,00h  ; символ Ж *
db  8ah,00h,00h,44h,82h,00h,92h,00h,92h,6ch;00h,00h     ; символ З *
db  8ah,00h,00h,0fah,04h,08h,10h,20h,40h,0beh,00h,00h   ; символ И *
db  8ah,00h,00h,0fah,04h,08h,90h,20h,40h,0beh,00h,00h   ; символ Й *
db  8bh,00h,00h,0feh,00h,10h,00h,28h,44h,82h,00h,00h    ; символ К *
db  8bh,00h,02h,3ch,40h,80h,00h,80h,00h,0feh,00h,00h    ; символ Л *
db  8ah,00h,00h,0beh,40h,20h,10h,20h,40h,0beh,00h,00h   ; символ М *
db  8ah,00h,00h,0feh,00h,10h,00h,10h,00h,0feh,00h,00h   ; символ Н *
db  8ah,00h,00h,7ch,82h,00h,82h,00h,82h,7ch,00h,00h     ; символ О *
db  8ah,00h,00h,0feh,00h,80h,00h,80h,00h,0feh,00h,00h   ; символ П *
db  8ah,00h,00h,0feh,00h,90h,00h,90h,00h,60h,00h,00h    ; символ Р *
db  8ah,00h,7ch,82h,00h,82h,00h,82h,00h,44h,00h,00h     ; символ С *
db  8ah,00h,00h,80h,00h,80h,7eh,80h,00h,80h,00h,00h     ; символ Т *
db  8ah,00h,0e2h,10h,02h,10h,02h,10h,02h,0fch,00h,00h   ; символ У *
db  8ah,00h,00h,70h,88h,00h,0feh,00h,88h,70h,00h,00h    ; символ Ф *
db  8bh,00h,00h,82h,44h,28h,10h,28h,44h,82h,00h,00h     ; символ Х *
db  8ah,00h,00h,0feh,00h,02h,00h,02h,0fch,03h,00h,00h   ; символ Ц *
db  8ah,00h,00h,0e0h,10h,00h,10h,00h,10h,0eeh,00h,00h   ; символ Ч *
db  8ah,00h,00h,0feh,00h,02h,0fch,02h,00h,0feh,00h,00h  ; символ Ш *
db  8ah,00h,0feh,00h,02h,0fch,02h,00h,0feh,03h,00h,00h  ; символ Щ *
db  8ah,00h,80h,00h,0feh,00h,12h,00h,12h,0ch,00h,00h    ; символ Ъ *
db  8ah,00h,0feh,00h,12h,00h,12h,0ch,00h,0feh,00h,00h   ; символ Ы *
db  8ah,00h,0feh,00h,12h,00h,12h,00h,12h,0ch,00h,00h    ; символ Ь *
db  8ah,00h,00h,44h,82h,10h,82h,10h,82h,7ch,00h,00h     ; символ Э *
db  8ah,00h,0feh,00h,10h,6ch,82h,00h,82h,7ch,00h,00h    ; символ Ю *
db  8ah,00h,62h,94h,08h,90h,00h,90h,00h,0feh,00h,00h    ; символ Я *
db  8ah,00h,04h,2ah,00h,2ah,00h,2ah,14h,02h,00h,00h     ; символ а *
db  8ah,00h,3eh,00h,2ah,00h,2ah,00h,2ah,04h,00h,00h     ; символ б *
db  8ah,00h,3eh,00h,2ah,00h,2ah,00h,2ah,14h,00h,00h     ; символ в *
db  8ah,00h,3eh,00h,20h,00h,20h,00h,20h,10h,00h,00h     ; символ г *
db  0ah,00h,00h,38h,45h,00h,45h,00h,45h,3ah,00h,00h     ; символ д *
db  8ah,00h,1ch,00h,2ah,00h,2ah,00h,2ah,10h,00h,00h     ; символ е *
db  8ah,00h,00h,22h,14h,08h,36h,08h,14h,22h,00h,00h     ; символ ж *
db  8ah,00h,00h,14h,22h,08h,22h,08h,22h,14h,00h,00h     ; символ з *
db  8ah,00h,3ch,02h,04h,08h,10h,20h,00h,3eh,00h,00h     ; символ и *
db  8ah,00h,3ch,02h,84h,08h,90h,20h,00h,3eh,00h,00h     ; символ й *
db  8ah,00h,3eh,00h,08h,00h,08h,04h,10h,22h,00h,00h     ; символ к *
db  8ah,00h,02h,0ch,10h,20h,00h,20h,00h,3eh,00h,00h     ; символ л *
db  8ah,00h,00h,1eh,20h,10h,08h,10h,20h,1eh,00h,00h     ; символ м *
db  8ah,00h,00h,3eh,00h,08h,00h,08h,00h,3ch,00h,00h     ; символ н *
db  8ah,00h,00h,1ch,22h,00h,22h,00h,22h,1ch,00h,00h     ; символ о *
db  8ah,00h,00h,3eh,00h,20h,00h,20h,00h,3eh,00h,00h     ; символ п *
db  0ah,00h,00h,7fh,00h,44h,00h,44h,00h,38h,00h,00h     ; символ р *
db  8ah,00h,1ch,22h,00h,22h,00h,22h,00h,22h,00h,00h     ; символ с *
db  8ah,00h,20h,00h,20h,1eh,20h,00h,20h,00h,00h,00h     ; символ т *
db  0ah,00h,79h,04h,01h,04h,01h,04h,01h,7eh,00h,00h     ; символ у *
db  0ah,00h,00h,38h,44h,00h,7fh,00h,44h,38h,00h,00h     ; символ ф *
db  8ah,00h,00h,22h,14h,08h,00h,08h,14h,22h,00h,00h     ; символ х *
db  8ah,00h,00h,3eh,00h,02h,00h,02h,3ch,03h,00h,00h     ; символ ц *
db  8ah,00h,30h,08h,00h,08h,00h,08h,00h,3eh,00h,00h     ; символ ч *
db  8ah,00h,00h,3eh,00h,02h,3eh,02h,00h,3eh,00h,00h     ; символ ш *
db  8ah,00h,3eh,00h,02h,3ch,02h,00h,3ch,03h,00h,00h     ; символ щ *
db  8ah,00h,20h,00h,3eh,00h,0ah,00h,0ah,04h,00h,00h     ; символ ъ *
db  8ah,00h,3eh,00h,0ah,00h,0ah,04h,00h,3eh,00h,00h     ; символ ы *
db  8ah,00h,00h,3eh,00h,0ah,00h,0ah,00h,04h,00h,00h     ; символ ь *
db  8ah,00h,22h,00h,22h,08h,22h,08h,22h,1ch,00h,00h     ; символ э *
db  8ah,00h,3eh,00h,08h,14h,22h,00h,22h,1ch,00h,00h     ; символ ю *
db  9ah,00h,10h,2ah,04h,28h,00h,28h,00h,3eh,00h,00h     ; символ я *
;
;
```

```
;= = = = = = = = = = = = = Стартовый блок = = = = = = = = = = = = = =
Begin:      push cs
            pop ds
            lea dx,cs:table     ; вывод на дисплей
            mov ah,09h          ; заголовка
            int 21h             ; программы
            xor bx,bx
            mov es,bx
            mov ax,es:[5ch]     ; получение и сохранение
            mov dx,es:[5eh]     ; адреса входа в
            mov cs:rem_17_off,ax  ; стандартный обра-
            mov cs:rem_17_seg,dx  ; ботчик прерывания INT17h
            lea dx,cs:new_int17
            mov es:[5ch],dx
            mov es:[5eh],cs
            call load           ; загрузка конфигурации
                                ; кириллицы в ОЗУ принтера
            lea dx,cs:Begin     ; выход с сохранением
            int 27h             ; в памяти
;
table       db 'Driver for printer Star LC-10',0ah,0dh
            db '(by Sharapov & Br.) Rostov/Don',0ah,0dh,'$'
;
command     ends
            end
```

Листинг 2: Пользовательский обработчик прерывания INT 17h для принтера EPSON LQ–500

```
;
;= = = = = = = = = = = = Секция обработки INT 17h = = = = = = = = = = =
;
new_int17   proc far
            push bx
            cmp ah,0h
            jnz nowork
            cmp al,80h      ; символ из правой части ?
            jb en           ; нет - переход на *en
            cmp al,0b0h     ; код символа меньше 176
            jb cont         ; да - переход на *cont
            cmp al,0e0h     ; код символа меньше 224
            jb en           ; да - переход на *en
            cmp al,0f1h     ; нет, код символа больше 241
            jg en           ; да - переход на *en
            sub al,176      ; уменьшить код символа на 176
            jmp Russian     ; переход на *Russian
cont:       sub al,128      ; уменьшить код символа на 128
            jmp Russian     ; переход на *Russian
en:         push ax
            call rom        ; переключение на работу с ПЗУ
            pop ax
            jmp nowork
Russian:    mov bx,ax       ; в BX - преобразованный код кириллицы
            mov al,byte ptr cs:[Decod+bx]
            test al,0ffh    ; символ-аналог существует?
            jnz en          ; да, переход на *en
            mov bx,ax       ; нет, получаем номер соответствующей
            mov al,byte ptr cs:[TabRam+bx]  ; ячейки ОЗУ
            push ax
            call ram        ; переключение на работу с ОЗУ
            pop ax
nowork:     pop bx
            xor dx,dx
            jmp dword ptr cs:int17_off  ; вызов старого вектора INT17h
new_int17   endp
;
Decod   db  41h,00h,42h,00h,00h,45h,00h,33h,00h,00h,4Bh,00h
        db  4Dh,48h,4Fh,00h,50h,43h,54h,00h,00h,58h,00h,00h
        db  00h,00h,00h,00h,00h,00h,00h,00h
        db  61h,00h,00h,00h,00h,65h,00h,00h,75h,75h,00h,00h
        db  00h,00h,6Fh,6Eh,70h,63h,6Dh,79h,00h,78h,00h,00h
        db  00h,00h,00h,00h,00h,00h,00h,00h
;
TabRam  db  00h,3Eh,00h,3Fh,40h,00h,41h,00h,42h,42h,00h,43h
        db  00h,00h,00h,44h,00h,00h,00h,45h,46h,00h,47h,48h
        db  49h,4Ah,4Bh,4Ch,4Dh,4Eh,4Fh,50h
        db  00h,51h,52h,53h,54h,00h,55h,56h,00h,00h,57h,58h
        db  59h,5Ah,00h,00h,00h,00h,00h,00h,5Bh,00h,5Ch,5Dh
        db  5Eh,5Fh,60h,61h,62h,63h,64h,65h
;
            db 'Sharapov & Br'
message     db 0dh,0ah,'Cyrillic for EPSON LQ-500',0dh,0ah,'$'
int17_off   dw ?            ; смещение старого обработчика INT 17h
int17_seg   dw ?            ; сегмент старого обработчика INT 17h
```

```
; ESC последовательность загрузки
определяемых пользователем символов
cod:        db 1bh,26h,00h,61,61,00h,07h,02h
; байты конфигурации символов кириллицы
cod1:
db  17h,0efh,0d0h,08h,10h,20h,10h,20h,10h,08h,10h,20h
db  10h,20h,10h,08h,18h,60h,10h,07h,80h                 ; <- Б
db  17h,0ffh,0f0h,08h,00h,00h,10h,00h,00h,08h,00h,00h
db  10h,00h,00h,08h,00h,00h,10h,00h,00h                 ; <- Г
db  00h,00h,1ch,03h,0ffh,0e0h,0ch,00h,10h,10h,00h,20h
db  08h,00h,10h,17h,0ffh,0e0h,00h,00h,1ch               ; <- Д
db  1fh,01h,0f0h,00h,0eeh,00h,00h,10h,00h,1fh,0efh,0f0h
db  00h,10h,00h,00h,0eeh,00h,1fh,01h,0f0h               ; <- Ж
db  1fh,0ffh,0d0h,00h,00h,20h,00h,00h,0c0h,00h,03h,00h
db  00h,0ch,00h,00h,10h,00h,1fh,0efh,0f0h               ; <- И
db  00h,00h,10h,00h,0ffh,0e0h,03h,00h,00h,06h,00h,00h
db  08h,00h,00h,10h,00h,00h,0fh,0ffh,0f0h               ; <- Л
db  17h,0ffh,0f0h,08h,00h,00h,10h,00h,00h,08h,00h,00h
db  10h,00h,00h,08h,00h,00h,17h,0ffh,0f0h               ; <- П
db  1fh,0e0h,00h,00h,10h,10h,00h,08h,20h,00h,10h,10h
db  00h,08h,60h,00h,11h,80h,1fh,0eeh,00h                ; <- У
db  03h,0feh,00h,04h,01h,00h,02h,02h,00h,1dh,0fdh,0f0h
db  02h,02h,00h,04h,01h,00h,03h,0feh,00h                ; <- Ф
db  1fh,0ffh,0d0h,00h,00h,20h,00h,00h,10h,00h,00h,20h
db  00h,00h,10h,1fh,0ffh,0e0h,00h,00h,1ch               ; <- Ц
db  1fh,0e0h,00h,00h,10h,00h,00h,08h,00h,00h,10h,00h
db  00h,08h,00h,00h,10h,00h,1fh,0efh,0f0h               ; <- Ч
db  1fh,0ffh,0d0h,00h,00h,20h,00h,00h,10h,1fh,0ffh,0e0h
db  00h,00h,10h,00h,00h,20h,1fh,0ffh,0d0h               ; <- Ш
db  1fh,0ffh,0d0h,00h,00h,20h,00h,00h,10h,1fh,0ffh,0e0h
db  00h,00h,10h,00h,00h,20h,1fh,0ffh,0dch               ; <- Щ
db  10h,00h,00h,08h,00h,00h,17h,0efh,0d0h,00h,10h,20h
db  00h,20h,10h,00h,10h,20h,00h,0fh,0c0h                ; <- Ъ
db  1fh,0efh,0d0h,00h,10h,20h,00h,20h,10h,00h,10h,20h
db  00h,0fh,0c0h,00h,00h,00h,1fh,0ffh,0f0h              ; <- Ы
db  1fh,0efh,0d0h,00h,10h,20h,00h,20h,10h,00h,10h,20h
db  00h,20h,10h,00h,10h,20h,00h,0fh,0c0h                ; <- Ь
db  04h,00h,40h,08h,00h,20h,10h,10h,10h,08h,20h,20h
db  10h,10h,10h,08h,20h,20h,07h,0dfh,0c0h               ; <- Э
db  1fh,0efh,0f0h,00h,10h,00h,07h,0efh,0c0h,08h,00h,20h
db  10h,00h,10h,08h,00h,20h,07h,0ffh,0c0h               ; <- Ю
db  07h,0e0h,10h,08h,10h,60h,10h,09h,80h,08h,16h,00h
db  10h,08h,00h,08h,10h,00h,17h,0efh,0f0h               ; <- Я
db  10h,7ch,0c0h,28h,80h,20h,25h,00h,40h,22h,80h,20h
db  11h,00h,40h,20h,80h,20h,10h,7fh,0c0h                ; <- б
db  00h,0fbh,0d0h,01h,04h,20h,00h,88h,10h,01h,04h,20h
db  00h,88h,10h,00h,74h,20h,00h,03h,0c0h                ; <- в
db  01h,7fh,0f0h,00h,80h,00h,01h,00h,00h,00h,80h,00h
db  01h,00h,00h,00h,80h,00h,01h,00h,00h                 ; <- г
db  00h,00h,1ch,00h,7fh,0e0h,00h,80h,10h,01h,00h,20h
db  00h,80h,10h,01h,0ffh,0e0h,00h,00h,1ch               ; <- д
db  01h,0c0h,70h,00h,31h,80h,00h,0a0h,00h,01h,0f5h,0f0h
db  00h,0a0h,00h,00h,31h,80h,01h,0c0h,70h               ; <- ж
db  00h,00h,40h,00h,80h,20h,01h,04h,10h,00h,88h,20h
db  01h,04h,10h,00h,8ah,20h,00h,71h,0c0h                ; <- з
db  01h,0ffh,0f0h,00h,04h,00h,00h,08h,00h,00h,04h,00h
db  00h,1bh,00h,00h,60h,0c0h,01h,80h,30h                ; <- к
db  00h,00h,10h,00h,00h,20h,00h,3fh,0c0h,00h,40h,00h
db  00h,80h,00h,01h,00h,00h,00h,0ffh,0f0h               ; <- л
db  01h,7fh,0f0h,00h,80h,00h,00h,60h,00h,00h,18h,00h
db  00h,60h,00h,00h,80h,00h,01h,7fh,0f0h                ; <- м
db  01h,0fbh,0f0h,00h,04h,00h,00h,08h,00h,00h,04h,00h
db  00h,08h,00h,00h,04h,00h,01h,0fbh,0f0h               ; <- н
db  00h,0ffh,0e0h,01h,00h,10h,00h,80h,20h,01h,7fh,0dfh
db  00h,80h,20h,01h,00h,10h,00h,0ffh,0e0h               ; <- ф
db  01h,0ffh,0d0h,00h,00h,20h,00h,00h,10h,00h,00h,20h
db  00h,00h,10h,01h,0ffh,0e0h,00h,00h,1ch               ; <- ц
db  01h,0f8h,00h,00h,04h,00h,00h,02h,00h,00h,04h,00h
db  00h,02h,00h,00h,04h,00h,01h,0fbh,0f0h               ; <- ч
db  01h,0ffh,0d0h,00h,00h,20h,00h,00h,10h,01h,0ffh,0e0h
db  00h,00h,10h,00h,00h,20h,01h,0ffh,0d0h               ; <- ш
db  01h,0ffh,0d0h,00h,00h,20h,00h,00h,10h,01h,0ffh,0e0h
db  00h,00h,10h,00h,00h,20h,01h,0ffh,0dch               ; <- щ
db  01h,00h,00h,00h,80h,00h,01h,7bh,0d0h,00h,04h,20h
db  00h,08h,10h,00h,04h,20h,00h,03h,0c0h                ; <- ъ
db  01h,0fbh,0d0h,00h,04h,20h,00h,08h,10h,00h,04h,20h
db  00h,03h,0c0h,00h,00h,00h,01h,0ffh,0f0h              ; <- ы
db  01h,0fbh,0d0h,00h,04h,20h,00h,08h,10h,00h,04h,20h
db  00h,08h,10h,00h,04h,20h,00h,03h,0c0h                ; <- ь
db  00h,40h,40h,00h,80h,20h,01h,08h,10h,00h,84h,20h
db  01h,08h,10h,00h,84h,20h,00h,7bh,0c0h                ; <- э
db  01h,0fbh,0f0h,00h,04h,00h,00h,7bh,0c0h,00h,80h,20h
db  01h,00h,10h,00h,80h,0c0h,00h,7bh,0c0h               ; <- ю
db  00h,78h,10h,00h,84h,60h,01h,02h,80h,00h,85h,00h
db  01h,02h,00h,00h,84h,00h,01h,7bh,0f0h                ; <- я
```

– процессы, выполняемые в фоновом режиме, при распечатывании текста осуществляются с некоторым замедлением;
– при выводе символов на принтер в графическом режиме некорректно отрабатываются изменения режимов печати ("PICA", "CONDENS" и так далее). Эта некорректность заключается в том, что начертания графических символов не подвергаются изменениям, характерным для смены режима печати, для символов, конфигурация которых хранится в ОЗУ или ПЗУ принтера.

Так что с учетом этих недостатков можно сделать вывод: графические возможности принтера — это хорошо, но при наличии достаточного объема ОЗУ в эксплуатируемом принтере лучше использовать загружаемые в него пользовательские шрифты.

И все же вернемся к принтерам, в которых без графики не удается осуществить их русификацию. Как и в случаях, описанных ранее, Ваша программа будет состоять из двух частей — рабочей и установочной.

Установочная часть программы претерпевает некоторые изменения, связанные с исключением из числа задач, возложенных на нее, проблемы загрузки знакогенератора в ОЗУ принтера. Она отвечает за переустановку используемого прерывания INT 17h (рис.4) и резидентное размещение рабочей части программы в памяти.

Из-за отсутствия или же неиспользования ОЗУ принтера соответствующим образом изменяется и рабочая часть программы. Блок-схема алгоритма рабочей части программы представлена на рис. 5. Она очень похожа на блок-схему рабочей части программы русификации принтера с ограниченным объемом ОЗУ. Перед тем как приступить к описанию новшеств, необходимо немного напомнить Вам, как печатаются символы в графическом режиме.

Команда (ESC последовательность), отвечающая за вывод на печать в графическом режиме, несет в себе

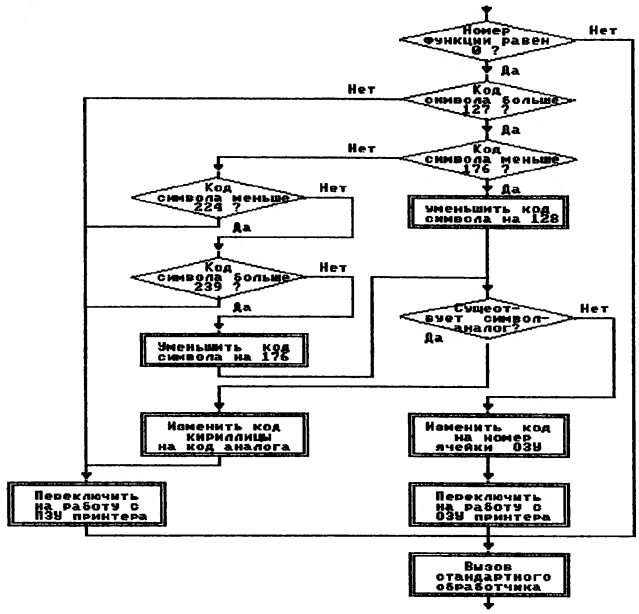


Рис. 4. Блок–схема алгоритма пользовательского обработчика прерывания INT 17h (EPSON LQ–500)

также информацию о ширине (исчисляемой числом колонок) этого символа и о плотности пропечатывания эти колонок, определяющих вкупе конфигурацию печатаемого символа. Формат команды вывода символа на принтер имеет следующий вид:

<ESC> "*" chr(m) chr(n1) chr(n2)
где m — код используемого графического режима;
n1, n2 — идентификаторы числа колонок, резервируемых для графического символа [5].

Игры и как с ними бороться

Пусть в меня бросят камень фэны компьютерных игр, ибо речь пойдет о них, об играх и как со всем этим можно бороться.

— А в чем, собственно, проблема, у нас что, играть запрещено?

Конечно же, нет. Но, по моему глубокому убеждению, игры приносят одним радость, другим чувство досады за "бесцельно прожитые годы".

Лично мне они приносили ежедневные неудобства тем, что пожирали львиную часть рабочего (да и нерабочего) времени. Бороться с этой заразой административными мерами не представляется возможным. Запретить играть невозможно, к тому же всегда найдется дядя, который обоснует, что работа у него суперважная, приоритетная и так далее, поэтому ему необходимо выделить все машинное время для выполнения этой архиважной задачи.

Знакомая ситуация?

Ничто человеческое мне не чуждо, но сколько можно ждать, когда наиграются все желающие? А что делать тем, кто в силу сложившихся обстоятельств не может повышать свою квалификацию из-за этих горе-пользователей?

Именно эти вопросы, вернее, поиски ответов на них, побудили меня заняться изучением языков программирования и всего того, что к этому прилагается.

Сначала был командный язык BAT файлов, затем Pascal, C. Было написано множество программ. В то время я и понятия не имел, что такое TSR программа. На сегодняшний день я не знаю лучшего лекарства, чем TSR, поэтому и предлагаю свой вариант решения этой проблемы. А дабы не утомлять читателя, перехожу к существу вопроса.

Итак, как обнаружить игру и бороться с ней?

В принципе, можно вписать в командный файл что-либо нехорошее, но это не слишком эстетично и практично, к тому же, надо рассчитывать на более грозного соперника. Можно смастерить какую-нибудь резидентную программу, которая бы через заданный промежуток времени говорила JMP на F000:FFF0 (перегрузка). А можно по таймеру проверять текущий видеорежим. Для этого достаточно с небольшой периодичностью просматривать ячейку в области данных BIOS по адресу 0000:0449h и, если обнаружена графика (игры, как правило, работают в графических режимах), предпринять какие-нибудь действия, например, принудительно заменить видеорежим на текстовый или просто сделать JMP в никуда. Настроение играющего будет не из лучших. Однако следует заметить, что при таком подходе к делу мы можем помешать работе "настоящего" пользователя, так как, работая с графикой (гра-

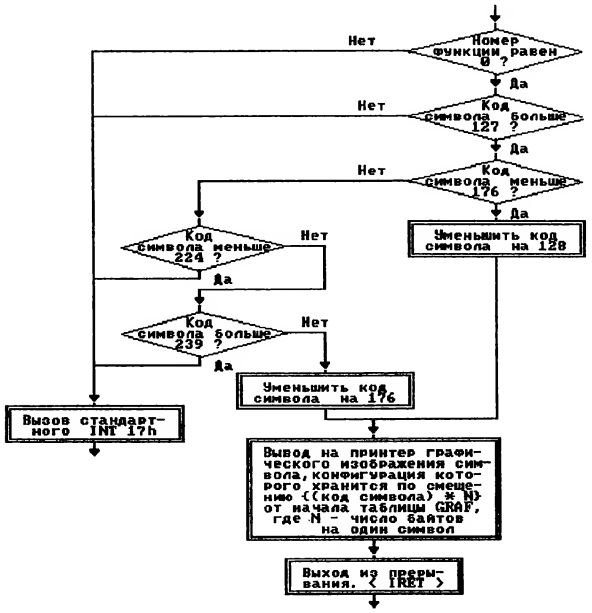


Рис. 5. Блок–схема алгоритма рабочей части программы EPSON_LX.ASM (EPSON LX–1050)

После приема последовательности на принтере следующие за ней (n2*256+n1) байтов интерпретируются как закодированная конфигурация выводимого на печать символа. Самая главная особенность этой графической команды заключается в том, что ее отработка принтером происходит немедленно.

Исходя из полученных сведений, можно сделать вывод, что создаваемый Вами пользовательский обработчик прерывания, отвечающего за вывод символа на печать (INT 17h), должен выполнять укрупненно следующее:

- разделять поток символов, выводимых на печать по признаку "кириллица" или "не кириллица";
- в случае получения символа "не кириллица" осуществлять переход к исполнению стандартного обработчика прерывания INT 17h;
- в случае получения символа "кириллица" осуществлять пересылку на принтер управляющей ESC последовательности, отвечающей за распечатывание в графическом режиме соответствующего символа.

На этом, в принципе, можно завершить обзор способов программной русификации матричных принтеров. Предлагаемый Вашему вниманию материал изложен в несколько сжатой форме, которая подразумевает, что Вы владеете некоторыми знаниями в области системного программирования. Материал также не является всеобъемлющим. Как Вы могли убедиться, некоторые фрагменты, более или менее специализированные, были в нем опущены, а также не были рассмотрены многочисленные вопросы, лежащие в сопредельных с рассматриваемой проблемой областях. ■

Литература

1. **Д. Брэдли. Программирование на языке ассемблера для персональной ЭВМ фирмы IBM.- М.: Радио и связь, 1988 г.**
2. **Орлин Шопов. Прекодиране на текст//Компютър за Вас. - 1990.- № 5-6.**
3. **Bedienungshandbuch LQ-500. EPSON Technologie, die Yeichen setzt. - Dusseldorf: EPSON Deutschland GmbH, 1987.**
4. **Multi-font LC-10. Users manual. - Star Micronics Co., Ltd., 1989.**
5. **Печатающее устройство LX-1050: Инструкция по эксплуатации.- Япония: SEIKO EPSON Corporation, 1988.**

фические редакторы, САПР и прочее), он невольно становится объектом атаки нашей программы. К тому же не следует доверять обработчику прерываний таймера (int 08h), так как очень многие игры перехватывают и самостоятельно работают с прерываниями таймера и клавиатуры, при этом они могут не отдавать управление старому вектору и, следовательно, наша программа никогда не получит управления.

Как же перехватить игру?

А, собственно, зачем? Давайте лучше встанем в цепочку обработчиков прерывания int 21h и будем внимательно изучать программы, которые должны быть загружены. То есть будем отбирать интересующие нас программы прямо на этапе загрузки. Для этого нам придется составить небольшую табличку сигнатур наиболее популярных игровых программ. Сделать это несложно. Например, находясь в Norton Commander, достаточно нажать кнопку <F3> на интересующем нас EXE или COM файле, установить клавишей <F8> режим просмотра HEX и при помощи какой-нибудь резидентной программы снять копию экрана в текстовый файл. Таким же образом нужно поработать со всеми интересующими нас программами. В полученных снимках экрана вы найдете цифры, готовые к употреблению, а заодно и комментарии, поясняющие их принадлежность. Именно таким образом был составлен файл GAMELIB.LBL (листинг 2), содержащий сигнатуру нескольких игр.

А теперь остается только набрать текст Game$.ASM (листинги 1, 3) без ошибок, оттранслировать Turbo Assembler (tasm GAME$.ASM, tlink /t/x/ GAME$.OBJ) и поставить программу на боевое дежурство. При появлении новых игр для их отбора достаточно вносить новую сигнатуру в исходный текст программы, транслировать его и заменять предыдущую версию свежей.

Некоторые соображения по возможной модификации программы: находясь в обработчике прерываний 21h, можно дополнительно проверять текущее время и в зависимости от этого разрешать или запрещать загрузку программы. Мой личный опыт подсказывает, что этого делать не следует, так как пользователю не запретишь работать с системным временем. Если Вы будете самостоятельно отрабатывать функцию "Set System Time", то пользователь просто войдет в CMOS и сделает то, что надо, хорошо, если он сделает это корректно, а если нет...? При небольшом числе сигнатур их лучше держать в теле программы (как сделано в этом варианте), а при наличии большого списка программ можно держать их в файле на диске и по мере необходимости подкачивать их в заранее рассчитанный и отведенный для этих целей буфер.

Еще одна мысль: можно уменьшить длину сигнатуры за счет дополнительных усилий, например, считывать DWORD или даже WORD по заранее вычисленному смещению, не забыв при этом проверить размер программы.

Как говорится, нет предела совершенству. А раз так, углубимся с пользой для себя в изучение ассемблера. Транслируем, модифицируем, в общем, вперед!

А.Аброськин ■

Листинг 1

```
;-----------------------------------------
; Резидентный перехватчик игр
;-----------------------------------------
; Game$.ASM was written by AVA Soft.
; Copyright (c) 1992
; Программа может быть использована
; в некоммерческих целях
; Автор: Аброськин Андрей Владимирович
;-----------------------------------------

INCLUDE     MACRO.REG

code        segment
            assume  cs:code, ds:code, es:code
            org 100h

start:      jmp  install        ; Переходим на установку программы
                                ;
presence    dw   670Fh          ; Ключевое слово присутствия
                                ; программы в памяти

Game_entry      proc far        ; Точка входа в программу
                                ; по прерыванию 21h
PushAll                         ; Сохраняем регистры (см. MACRO.REG)
            cmp ax,4b00h        ; Это функция "Exec"?
            je  short Exec      ; Да, работаем с ней
            jmp return          ; Нет, выходим из обработчика
;---------------------------;
Exec:
; <DS,DX> уже установлены     ; Делаем попытку
; на то, что нам нужно        ; открыть файл, запущенный
            mov ax,3d00h        ; на выполнение
            pushf               ;
            call  dword ptr cs:[old21o]   ; В случае ошибки
            jc    return        ; передаем ее старому вектору
            mov cs:handle,ax
            mov bx,cs:handle    ; В случае успеха заносим file
            mov cx,0            ; handle в BX и устанавливаем
            mov dx,0            ; указатель файла на начало
            mov ax,4200h        ;
            pushf               ;
            call  dword ptr cs:[old21o]   ; Инструкция int 21h
            jc    return
            mov bx,cs:handle    ;
            mov cx,cs           ;
            mov ds,cx           ; Устанавливаем буфер
            lea dx,buffer       ; и считываем в него
            mov cx,10           ; первые 10 байтов
            mov ah,3Fh          ; исследуемой программы
            pushf               ;
            call  dword ptr cs:[old21o]   ; Инструкция int 21h
            jc    return
            mov bx,cs:handle
            mov ah,3Eh          ; Закрываем файл
            pushf               ;
            call  dword ptr cs:[old21o]   ; Инструкция int 21h
            jc    return
;---------------------------     ; Устанавливаем в BX число
            mov bx,24           ; строк таблицы сигнатур
            mov cx,cs           ;
            mov es,cx           ; Настраиваем ES и DS
            mov ds,cx           ;
            lea di,cs:[signat]  ; Загружаем смещение сигнатур
            mov dx,di           ; и запоминаем его в DX
            lea si,cs:[buffer]  ; [SI] указывает на сигнатуру
                                ; исследуемой программы
            cld                 ; Сбрасываем флаг направления
loop$:
            mov cx,10           ; Сканируем 10 элементов таблицы
            repe cmpsb          ; сигнатур на предмет совпадения,
            je   found          ; если нашли - выходим из цикла
            add  dx,10          ;
            mov di,dx           ; Корректируем смещение в таблице
            mov ax,cs           ; Устанавливаем указатель buffer
            mov ds,ax           ; на начало
            lea si,cs:[buffer]  ;
            dec  bx             ; Повторяем цикл, пока в BX
            jne  loop$          ; еще что-то есть, если ничего
;---------------------------     ; хорошего не нашли, то
return:
            PopAll              ; восстанавливаем регистры и
                                ; передаем управление на
            db   0EAh           ; инструкцию far jump по
old21o      dw   0              ; старому адресу обработчика int 21H
old21s      dw   0              ;
                                ;
found:                          ;
            PopAll              ; Восстанавливаем регистры,
            stc                 ; выставляем флаг и код
            mov ax,10h          ; ошибки ("Cannot execute file")
            ret  2              ;
                                ;
Game_entry  endp                ;
;---------------------------     ;
handle      dw   (?)            ; handle файла
buffer      db   10 dup (?)     ; Буфер для временного хранения
                                ; сигнатуры вызываемой программы
INCLUDE     GAMELIB.LBL         ; Таблица сигнатур
;-----------------------------------------
; Транзитная часть программы Game$.COM
;-----------------------------------------
install:                        ;
            mov ax,3521h        ; Получаем адрес обработчика
            int  21h            ; прерывания 21H
            mov ax,es:[bx-2]    ; Проверяем ключевое слово
            cmp ax,cs:presence  ; Если не обнаружено, то
            jnz  load           ; загружаем в память,
            jmp  already        ; иначе игнорируем вызов

load:
            push es             ; Сохраняем регистры для
            push ds             ; светлого будущего
                                ;
            mov ax,3521h        ; Получаем адрес вектора 21H,
            int  21h            ; сегмент и смещение
            mov word ptr old21s,es
            mov word ptr old21o,bx
                                ;
                                ;
            lea  dx,Game_entry  ; Устанавливаем вектор 21H
            mov ax,2521h        ; на наш обработчик
            int  21h            ; прерывания
                                ;
            pop  ds             ; Восстанавливаем регистры,
            pop  es             ;
                                ; "терминируем" и говорим,
            mov dx,offset install   ; что программа будет резидентной
            int  27h            ;

already:
            mov al,ds:[082h]    ; Проверяем командную строку
            cmp al,'/'          ; и ищем ключ для выгрузки
            je   key_1          ; программы из памяти
            jmp  bad_key
key_1:
            mov al,ds:[083h]
            and  al,11011111b   ; Переводим символ в верхний
            cmp al,'E'          ; регистр
            je   key_2
            jmp  bad_key
key_2:
            mov al,ds:[084h]
            cmp al,'$'
            je   key_3
            jmp  bad_key
key_3:
            mov al,ds:[085h]
            cmp al,'$'
            je   unload
            jmp  bad_key

unload:
            mov ax,word ptr es:old21o   ; Получаем адрес для
            mov dx,word ptr es:old21s   ; восстановления вектора INT21H
            mov cx,es           ; Сохраняем резидентную
                                ; CS часть программы в CX,
            cli                 ; запрещаем прерывания
            xor  bx,bx          ;
            mov es,bx           ; Заносим в ES нулевой
            mov bx,21h*4        ; сегмент и записываем в
            mov es:[bx],ax      ; таблицу векторов украденный
            mov es:[bx+2],dx    ; адрес вектора  INT21H,
            sti                 ; разрешаем прерывания
                                ;
            mov ah,49h          ;
            mov es,cx           ; Восстанавливаем ES
            int  21h            ; и освобождаем память,
                                ; занятую нашей программой
bad_key:                        ;
            mov ax,4C00h        ; Выходим в DOS
            int  21h            ;
code        ends                ;
            end  start          ;
;-----------------------------------;
```

Листинг 2

```
;-------------------------------------------------------
; Файл GAMELIB.LBL для программы Game$.ASM
; Таблица сигнатур
; Copyright (c) 1992 by AVA Soft
;-------------------------------------------------------

signat db   4Dh,5Ah,02h,01h,03h,00h,00h,00h,02h,00h        ; golf.exe
       db   4Dh,5Ah,0DBh,01h,5Dh,01h,00h,00h,02h,00h       ; lhx.exe
       db   4Dh,5Ah,0FDh,00h,9Bh,00h,00h,00h,20h,00h       ; cfs.exe
       db   4Dh,5Ah,21h,00h,0D3h,00h,00h,00h,20h,00h       ; gpega.exe
       db   4Dh,5Ah,0E1h,00h,1Ah,00h,02h,00h,20h,00h       ; poker.exe
       db   4Dh,5Ah,0D0h,00h,3Dh,00h,1Bh,02h,89h,00h       ; tenis.exe
       db   0E9h,79h,2Ch,90h,90h,0CDh,0ABh,43h,6Fh,70h    ; tet.com
       db   0BCh,0B8h,07h,0BBh,7Ch,00h,0B4h,4Ah,0CDh,21h; f117.com
       db   4Dh,5Ah,44h,01h,29h,00h,00h,00h,02h,00h       ; another.exe
       db   4Dh,5Ah,62h,00h,54h,00h,3Bh,00h,20h,00h       ; columns.exe
       db   4Dh,5Ah,06h,00h,8Ah,00h,1Dh,00h,20h,00h       ; bl.exe
       db   4Dh,5Ah,0D7h,01h,0A3h,00h,00h,00h,20h,00h     ; chess.exe
       db   4Dh,5Ah,0FDh,00h,9Bh,00h,00h,00h,20h,00h      ; girls\a.exe
       db   4Dh,5Ah,04h,00h,9Fh,00h,00h,00h,02h,00h       ; diceclnd.exe
       db   4Dh,5Ah,78h,01h,0C1h,00h,0E5h,05h,80h,01h     ; pref\pc.exe
       db   4Dh,5Ah,0C7h,01h,0F1h,00h,00h,00h,20h,00h     ; prince.exe
       db   4Dh,5Ah,2Fh,01h,02h,00h,01h,00h,20h,00h       ; crazy.exe
       db   4Dh,5Ah,0DBh,01h,38h,00h,00h,00h,20h,00h     ; rockford.exe
       db   4Dh,5Ah,30h,01h,09h,00h,1Fh,00h,0Ah,00h      ; rockf.exe
       db   4Dh,5Ah,2Fh,01h,02h,00h,01h,00h,20h,00h      ; crazy.exe
       db   0E9h,14h,05h,06h,1Fh,0B9h,70h,0Dh,0BFh,00h   ; snake.com
       db   4Dh,5Ah,90h,00h,42h,00h,51h,01h,60h,00h      ; tetrill.exe
       db   4Dh,5Ah,0ACh,00h,0F1h,00h,00h,00h,20h,00h    ; tower.exe
       db   4Dh,5Ah,0A0h,00h,64h,00h,95h,00h,40h,00h     ; welltr.exe
```

Листинг 3

```
;-------------------------------------
;
; MACRO.REG для Game$.ASM
;
;-------------------------------------

PushAll  MACRO
         pushf
         push  ax
         push  bx
         push  cx
         push  dx
         push  di
         push  si
         push  bp
         push  es
         push  ds
         ENDM
;-------------------------------------
PopAll   MACRO
         pop   ds
         pop   es
         pop   bp
         pop   si
         pop   di
         pop   dx
         pop   cx
         pop   bx
         pop   ax
         popf
         ENDM
```

Анонс

"Монитор" версия 3.93

А.Штрик. Зарубежные CASE средства создания программного обеспечения в отечественных разработках
Рассмотрены зарубежные инструментальные средства автоматизированной разработки прикладного программного обеспечения, которые на официальных основаниях распространяются в России. Для каждого инструментального средства приводится его назначение и область применения, состав и основные возможности, а также аппаратная платформа функционирования.

А. Колесов, О. Павлова. Профессиональная работа в среде Microsoft BASIC
Часть 2. Работа с инструментальными библиотеками подпрограмм
Возможность разработки сколько-нибудь большой программы во многом определяется наличием готовых библиотек подпрограмм. А класс программиста — умением сформировать свой собственный набор инструмента и эффективно использовать его. При работе в системе QuickBASIC эта проблема имеет целый ряд особенностей по сравнению с традиционными компиляторами.

С. Никоненко. Программируем СУБД на Turbo Vision
Введение в объектно-ориентированную библиотеку DATA VISION для Turbo Pascal версии 6.0+. Приведен исходный текст конвертера файлов dBASE в файл СУБД PARADOX.

А. Пустеленин. CLIPPER и ООП: фантазия или реальность?
В статье предпринимается попытка применить возможности объектно-ориентированного программирования при создании программ на Clipper 5.0.

А. Тихонов. И надпись написал...
Рассматривается задача вывода на экран текста, подвергнутого произвольным линейным преобразованиям (перенос, изменение масштаба, вращение).

Л. Ливеровский. Немного о дисассемблерах. (Опыт работы с программой SOURCER)
В статье рассказывается о принципах работы дисассемблера SOURCER и приводится список его команд.

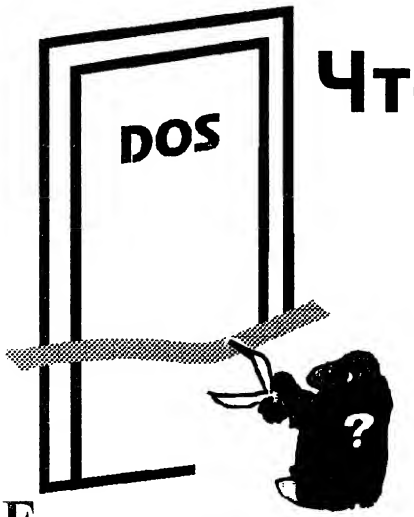
Ю. Косивцов. Двухкомпонентная антивирусная система
Описаны принципы построения резидентных антивирусных программ, блокирующих несанкционированный доступ к вычислительным ресурсам.

Д. Бройтман. Микропроцессор Pentium
С точки зрения программиста рассматривается новое поколение микропроцессоров фирмы Intel.

В. Нефедов, Д. Самсонов. Система геометрического моделирования CADKEY
CADKEY — графическая система конструкторско-проектных работ для персональных компьютеров. Является альтернативой хорошо известной российскому читателю графической системы AUTOCAD фирмы Autodesk.

М. Корсаков. Юридическая защита программ в России: практическая реализация законодательства
В статье рассматриваются основные положения Закона РФ "О правовой охране программ для электронно-вычислительных машин и баз данных" и проблемы практической реализации правовых норм по охране программ.

Игры с несколькими участниками
Часть 2
Вторая часть обзора, посвященного играм жанра Role Playing Games.



Что же там, в загрузочном секторе?

В. Аношин

Есть в любой ОС очень интересная часть — загрузочный код. Этот код хранится в загрузочном секторе диска. Интерес состоит в том, что эта очень маленькая программа (ее размер не превышает размер сектора) должна уметь очень много. Основная задача загрузочного кода — найти на загрузочном диске ядро операционной системы и, загрузив его в память, передать ему управление. Давайте посмотрим на загрузочный код DOS. Для выполнения своей задачи он должен:

1) определить, с какого диска производится загрузка системы;

2) на основании данных из таблицы параметров диска, которая является частью кода, определить расположение в секторах корневого каталога;

3) найти элемент каталога, соответствующий ядру ОС (файл IBMBIO.COM) и запомнить ссылку на первый кластер соответствующего файла;

4) найти и прочитать таблицу распределения файлов (FAT);

5) на основании информации из FAT считать в память файл ядра ОС и передать ему управление.

Получается, код загрузки должен выполнить в упрощенном виде функцию запуска процесса в ОС. Первая из приводимых программ представляет собой исходный текст такого кода (листинг 1). Она не является восстановленным загрузочным кодом DOS, но выполняет почти те же действия. Первоначально программа была написана на С, но затем с целью уменьшения размера кода была переведена на язык ассемблера (использовался Turbo Assembler версии 2). С той же целью были оставлены переменные в стеке. Компилируется следующими командами:

```
tasm boots.asm
tlink /t boots
```

Получаемый в результате компиляции com файл записывается на отформатированную дискету соответствующего объема на место оригинального загрузочного сектора любой программой типа DiskEditor из пакета Norton Utilities. В качестве "ядра" можно использовать вторую программу (листинг 2), которая после компиляции с теми же ключами, что и первая, копируется как обычный файл на подготовленную дискету.

Стартовая часть программы нужна для того, чтобы обеспечить независимость кода от смещения в памяти, что дает возможность собирать ее как обычную com программу. Так как конкретные значения элементов таблицы параметров диска зависят от типа носителя, то перед компиляцией программы нужно удостовериться в их соответствии типу диска, на который Вы собираетесь ее установить. Для этого можно использовать таблицу.

Таблица

Объем	360K	720K	1.2Mb	1.4Mb
Секторов на дорожку	9	9	15	18
Дорожек	40	80	80	80
Элементов корневого каталога	112	112	224	224
Секторов в FAT	2	3	7	9
Секторов на диске	720	1440	2400	2880
Секторов на кластер	2	2	1	1

Можно также воспользоваться программой DiskEdit для определения параметров диска.

Теперь о работе программы. Первым делом она пытается, читая по одному секторы корневого оглавления, найти имя файла IBMBIO.COM. Корневое оглавление начинается сразу за второй копией FAT на диске и имеет длину в секторах, равную числу элементов оглавления, деленному на размер сектора. Если поиск успешен, код загрузки запоминает ссылку на первый кластер этого файла и пытается прочитать в память по взятому с потолка адресу всю таблицу распределения файлов. Конечно, можно было бы читать только те секторы FAT, которые содержат необходимые на данный момент элементы, но тогда придется вводить в программу дополнительную логику обработки FAT, что сильно увеличит размер кода. Смещение FAT в секторах вычисляется как сумма чисел спрятанных и резервных секторов диска. Все остальные моменты, необходимые для понимания работы программы, освещены в комментариях в тексте самой программы. Кроме того, оставлены некоторые исходные операторы на С в виде комментариев.

Для получения более подробной информации о структуре диска в DOS читайте книгу Р.Джордейна "Справочник программиста ПК типа IBM PC, XT и AT". ■

Листинг 1

```
.MODEL TINY
;       СТРУКТУРА КОРНЕВОГО СЕКТОРА ДИСКА
boot_sect   STRUC
nam         DB   'IBM    5.0' ; Имя компании и версия системы
sectsiz     DW   512          ; Число байтов на сектор
clastsiz    DB   1            ; Число секторов на кластер
ressecs     DW   1            ; Число резервных секторов перед FAT
fatcnt      DB   2            ; Число FAT
rootsiz     DW   224          ; Максимальное число 32-байтовых
                              ; элементов оглавления
totsecs     DW   2880         ; Общее число секторов на носителе или в
                              ; разделе DOS
media       DB   0F0h         ; Дескриптор носителя (1-й байт FAT)
fatsize     DW   9            ; Число секторов в одной FAT
trksecs     DW   18           ; Число секторов на дорожку
headcnt     DW   2            ; Число головок чтения-записи
hidncnt     DD   0            ; Число спрятанных секторов
totsecsd    DD   0            ; Число секторов на диске,
                              ; если размер его > 32 Мбайтов
drive       DB   0            ;
res         DB   0
extBPB      DB   ')'
ser_nom     DD   0
vollabel    DB   'NEW DOS 0.0'
sysid       DB   '           '
boot_sect   ENDS
.DATA
_osname     DB   'IBMBIO  COM',0
_msg1       DB   0Ah,0Dh,'>',0
_msg2       DB   'Error!',0
_msg3       DB   'No kernel!',0
.CODE
            org  100h
LOCAL       _rootcnt,_rootbeg,_clstr,_fat:dword,_sect_l:dword,
            _buff:dword,_sect:byte,_trk,_head:byte,_mul,_eof=L
main        PROC
            jmp  SHORT    fp
            nop
_boot       boot_sect  < >
fp:
            ;jmp  far   07B0h:b
            DB   0EAh
            DW   OFFSET b,07B0h
b:
            ;Установка регистров для стека
            xor  ax,ax
            mov  ss,ax
            mov  bp,7C00h
            mov  sp,(7C00h-L-2)
            push cs
            pop  ds
            ;
            ; rootcnt = boot.rootsiz/16;
            ;
            mov  ax,_boot.rootsiz
            mov  cl,4
            shr  ax,cl
            mov  _rootcnt,ax
            ;
            ; rootbeg = boot.ressecs + boot.fatsize*boot.fatcnt;
            ;
            mov  al,_boot.fatcnt
            xor  ah,ah
            mul  _boot.fatsize
            add  ax,_boot.ressecs
            mov  _rootbeg,ax
            ;
            ;Чтение корневого каталога
            ; buff = (void far*)0x00700000;
            ;
            xor  ax, ax
            mov  WORD PTR _buff+2,0070h
            mov  WORD PTR _buff,ax
            ;
            ; for (_AX = 0; _AX < rootcnt; _AX++){
            ;
@6:
            push ax
            ;
            ; sect_l = _AX + rootbeg;
            ;
            add  ax,_rootbeg
            xor  dx, dx
            mov  cx,1
            ;
            ; Чтение очередного сектора корневого каталога
            ; read_ls();
            ;
            call read_ls
            sub  WORD PTR _buff,512
            ;
            ; Поиск имени файла ядра
            ; for(_AX = 0; _AX < 512; _AX+ = 32)
            ; If(memcmp(osname, &buff[_AX], 11) = = 0)goto @8;
            ;
            xor  ax,ax
@7:
            mov  cx,11
            les  di,_buff
            add  di,ax
            lea  si,_osname
            cld
            repe cmpsb
            jcxz @8
        add ax,32
            cmp  ax,512
            jb   @7
            ;
            ; };
            ;
            pop  ax
            inc  ax
            cmp  ax,_rootcnt
            jb   @6
            ;
            ; Не найден в корневом каталоге
            ; Перезагрузка системы
_ex:
            ; putstr('No kernel!');
            ;
            lea  si,_msg3
            call putstr
@14:
            jmp  @14
            ;
@8:
            pop  bx
            ;
            ; clstr = fn[_AX].clstrno;
            ;
            les  bx,_buff
            add  bx,ax
            mov  ax,WORD PTR es:[bx+26]
            mov  _clstr,ax
            ;
            ; putstr('>');
            ;
            lea  si,_msg1
            call putstr
            ;
            ; Чтение FAT
            ; buff = fat = (void far*)0x08000000;
            ;
            mov  dx,0800h
            xor  ax,ax
            mov  WORD PTR _fat+2,dx
            mov  WORD PTR _fat,ax
            mov  WORD PTR _buff+2,dx
            mov  WORD PTR _buff,ax
            ;
            ; sect_l = boot.ressecs
            ;
            mov  ax,_boot.ressecs
            xor  dx,dx
            mov  cx,_boot.fatsize
            ;
            ; read_ls();
            ;
            call read_ls
            ;
            ; Установка переменных для чтения FAT
            ; Для 12-битовой FAT
            ; mul = 3; eof = 0x0FF8;
            ;
            mov  _mul,3
            mov  _eof,0FF8h
            ;
            ; if(boot.totsecs > 0x5104 | | boot.totsecs = = 0){
            ;
            mov  ax,_boot.totsecs
            cmp  ax,5104h
            ja   @3
            or   ax,ax
            jnz  @4
            ;
            ; Для 16-битовой FAT
            ; mul = 4;eof = 0xFFF8;
            ;
@3:
            mov  _mul,4
            mov  _eof,0FFF8h
            ;
            ; };
            ;
@4:
            ;
            ; Чтение файла ОС
            ; buff = 0x00700000
            ;
            mov  WORD PTR _buff+2,0070h
            mov  WORD PTR _buff,0
@9:
            ;
            ; while(clstr < eof){
            ;
            mov  ax,_clstr
            cmp  ax,_eof
            jae  @13
            ;
            ; read_clstr();
            ;
            call read_clstr
            ;
            ; Вычисление номера следующего кластера
            ; _AX = fat[ (clstr*mul)/2 ];
```

Листинг 1 (продолжение)

```
                ;
                mov   ax,_clstr
                mul   _mul
                shr   ax,1
                les   si,_fat
                add   si,ax
                mov   ax,es:[si]
                ;
                ; If (_mul = = 3) clstr = (clstr%2) ? (_AX & 0x0FFF) : (_AX > > 4);
                ;
                cmp   _mul,3
                jne   @12
                mov   cl,4
                test  _clstr,1
                jne   @11
                shl   ax,cl
@11:
                shr   ax,cl
                ;
                ; else clstr = _AX;
                ;
@12:
                mov _clstr,ax
                ;
                ; };
                ;
                jmp   @9
@13:
                ;
                ; Передача управления прочитанному ядру
                ; goto system;
                ;
                DB    0EAh
                DW    0000h,0070h
main            ENDP
; Процедура чтения логического сектора
; ax:dx - номер логического сектора
read_ls         PROC
                mov   WORD PTR _sect_l,ax
                mov   WORD PTR _sect_l + 2,dx
@47:
                push  cx
                ;
                ; _CX = 3
                ; do{
                ;
                mov   cx,3
@48:
                push  cx
                ;
                ; sect_l+ = boot.hidncnt;
                ; _BX = sect_l / boot.trksecs;
                ; sect = sect_l % boot.trksecs + 1;
                ;
                mov   ax,WORD PTR _sect_l
                mov   dx,WORD PTR _sect_l + 2
                add   ax,WORD PTR _boot.hidncnt
                adc   dx,WORD PTR _boot.hidncnt+2
                div   _boot.trksecs
                mov   bx,ax
                inc   dl
                mov   _sect,dl
                ;
                ; trk = _BX / boot.headcnt;
                ;
                xor   dx,dx
                div   _boot.headcnt
                mov   _trk,ax
                ;
                ; _CX = trk < < 8 + (trk > > 2)&0x00C0 + _sect
                ;
                mov   ch,al
                shr   ax,1
                shr   ax,1
                mov   cl,al
                and   cl,0C0h
                add   cl,_sect
                ;
                ; head = _BX - (trk * boot.headcnt);
                ;
                mov   ax,_trk
                mul   _boot.headcnt
                sub   bx,ax
                ;
                ; _DH = head
                mov   dh,bl             ;Головка
                ;
                ; _DL = boot.drive
                mov   dl,_bpot.drive    ;Диск
                ;
                ; _AX = 0x0201
                mov   ax,0201h          ;Чтение &количество секторов
                ;
                ; _BX = (int)buff
                les   bx,_buff          ;Буфер
                int   13h
                ;
                ; } until( (CarryFlag = = 0) | | (--_CX = = 0) );
                ;
                pop   cx
                jnc   SHORT @51
                loop  @48
@49:
                ; Вывод сообщения об ошибке
                lea   si,_msg2
                call  putstr
@50:
                jmp   @50
@51:
                inc   WORD PTR _sect_l
                adc   WORD PTR _sect_l + 2,0
                add   WORD PTR _buff,512
                pop   cx
                loop  @47
                ret
read_ls         ENDP
; Процедура чтения кластера
; ax - номер кластера
read_clstr      PROC
                ;
                ; sect_l = rootbeg+rootcnt+(clstr-2)*boot.clastsiz;
                ;
                sub   ax,2
                mov   dl,_boot.clastsiz
                xor   dh,dh
                mul   dx
                add   ax,_rootbeg
                adc   dx,0
                add   ax,_rootcnt
                adc   dx,0
                mov   cl,_boot.clastsiz
                xor   ch,ch
                ;
                ; read_ls();
                ;
                call  read_ls
                ret
read_clstr      ENDP
putstr          PROC
                lodsb                   ;Загрузка в al выводимого символа
                cmp   al,00h            ;0 - конец строки
                je    M8
                mov   bx,7
                mov   ah,0Eh            ;Функция вывода символа на экран
                int   10h
                jmp   SHORT     putstr
M8:
                ret
putstr          ENDP
                end   main
```

Листинг 2

```
.MODEL      TINY
.DATA
str1        DB    0Ah,0Dh,"New DOS Version 0.0",0
.CODE
            org   100h
start       PROC
            call  a
a:
            mov   bp,sp
            mov   ax,ss:[bp]
            mov   bx,OFFSET a
            mov   cl,4
            cmp   ax,bx
            ja    @1
            sub   bx,ax
            shr   bx,cl
            mov   ax,cs
            sub   ax,bx
            jmp   @2
@1:
            sub   ax,bx
            shr   ax,cl
            mov   bx,cs
            add   ax,bx
@2:
            push  ax
            mov   ax,OFFSET b
            push  ax
            retf
b:
            mov   ax,cs
            mov   ds,ax
            mov   es,ax
            mov   si,OFFSET str1
            call  putstr
@3:
            jmp   @3
start       ENDP
putstr      PROC
            lodsb                   ;Загрузка в al выводимого символа
            cmp   al,00H            ;0 - конец строки
            je    @4
            mov   bx,7
            mov   ah,0EH            ;Функция вывода символа на экран
            int   10H
            jmp   SHORT     putstr
@4:
            ret
putstr      ENDP
end         start
```

Компания "Статус" — первое знакомство

Компания "Статус" существует всего около полутора лет — с осени 1991 года. Компания осуществляет поставку персональных компьютеров, установку локальных вычислительных сетей, сервисное и гарантийное обслуживание вычислительной техники. За короткий срок фирма сумела занять достаточно прочное место на рынке: объем продаж за 1992 год составил 1.5 млн. долларов США. "Статус" поставляет в Россию и страны СНГ комплектующие, отдельные блоки и мониторы производства STI Inc. (Калифорния, США) и Inbount Trade Inc. (Массачусетс, США). Компания "Статус" большое внимание уделяет подбору своих зарубежных партнеров, полагая, что гораздо выгоднее продвигать на российский рынок продукцию известных производителей, отличающуюся высокими техническими характеристиками и надежностью. Примером может служить персональный компьютер известной в нашей стране фирмы DEC (табл.). Хочется отметить отличное конструктивное исполнение (корпус типа slim-line с экранированием радиочастотных помех, сборка на заводах Digital Equipment Corp.), неплохое соотношение стоимости и технических характеристик модели DECstation 320SX (программно и аппаратно совместимой с IBM PC/AT), что выгодно отличает ее от собратьев из Юго-Восточной Азии. Моделью заинтересовались и представители многих государственных организаций. Причина понятна: низкий уровень паразитарных электромагнитных излучений, традиционно выдерживаемый фирмой DEC, позволяет использовать DECstation 320SX для хранения данных, составляющих коммерческую тайну.

Кроме того, "Статус" осуществляет продажу компьютеров собственной сборки. В течение 1992 года компания собрала (по конфигурации, определяемой заказчиком) более 1200 машин. Точкой отсчета при составлении контракта являются шесть моделей, краткие характеристики которых приведены далее.

8088/8087/640Кб/1,44/20-40Мб/1P+2S/VGA
80286-20/80287/1Мб/1,2+1,44/52Мб/1P+2S/VGA
80386SX-25/387SX/2Мб/1,2+1,44/80Мб/1P+2S/SVGA
386DX-40/387/4Мб/1,2+1,44/120Мб/1P+2S/SVGA
486SX-33/4-8Мб/1,2+1,44/120-340Мб/1P+2S/SVGA
486DX-50/4-8Мб/1,2+1,44/120-340Мб/1P+2S/SVGA

Если же у Вас возникла потребность в портативных средствах, "Статус" и в этом случае придет Вам на помощь. Большой популярностью пользуется у покупателей портативный компьютер с характеристиками:

80386SX-20/1-5Мб/1,44/20Мб/1P+2S/VGA-64Кб

Остается добавить только габариты: 210 * 297 * 47 мм и вес (с винчестером и аккумуляторами): 3,2 кг.

Компания "Статус" поставляет в комплекте с компьютерами принтеры (по выбору заказчика) производства фирм: Star, MicroLine, EPSON, Hewlett Packard.

Кроме того, Вы можете заказать как ставшие уже достаточно популярными модемы, сетевые платы, сканеры, так и достаточно экзотические для нашего рынка накопители CD-ROM.

Но и это еще не все. "Статус" поставляет "под ключ" издательские системы на базе PC:

386/486 + HP Laser Jet + HP Scan Jet + математическое обеспечение.

Наряду с локальными вычислительными сетями "Статус" поставляет и школьные учебные классы, отличающиеся высокими техническими характеристиками. Достаточно указать, что в число пользователей учебных классов компании "Статус" входит Военно-Воздушная Академия им. Жуковского.

Благоприятное впечатление остается и после знакомства с организацией сервисного и гарантийного обслуживания продукции компании "Статус". Так, все поставляемые фирмой изделия имеют гарантийный срок работы один год. Гарантийный ремонт выполняется, как правило, на месте эксплуатации (по итогам 1992 года гарантийное обслуживание потребовалось примерно 3% поставленных компьютеров). Это означает, что компьютер, вышедший из строя в Барнауле или Омске, совершенно не нужно возвращать для ремонта на "завод-изготовитель". Для решения этой нелегкой в условиях России задачи компания "Статус" заключила договоры о взаимном сервисном обеспечении с партнерами во многих городах России.

С некоторыми из фирм, в основном за пределами России, имеется еще более тесный контакт, который значительно сокращает срок получения аппаратуры. Продукцию компании "Статус" Вы можете заказать в фирмах "Пайтекс" (Киев, Украина), STT или Lietuvos Spektras (Вильнюс, Литва), Erla (Рига, Латвия), ЛОЗКОМ (Минск, Беларусь), и она будет немедленно поставлена со складов указанных фирм.

Познакомиться с продукцией фирмы "Статус" Вы сможете на выставке "Комтек-93" (стенд № 103).

Телефоны: (095) 213-07-47, (095) 213-17-27, (095) 462-38-16.

Факс: (095) 212-21-72, (095) 462-38-16 ■

Таблица

Основные характеристики DECstation 320SX

Модель	ER-PC-433-UJ-002
Процессор	i80386SX
Оперативная память	1–5 Мб
Гнездо для 80387SX	есть
Видеоадаптер	SVGA, 512 Кб
ГМД	3,5", 1,44 Мб
НЖМД	60 Мб, 17 ms
Свободные гнезда	3 * 16 битов
Шина	ISA bus
Тактовая частота шины	8 МГц
Габаритные размеры	382 * 394 * 107 мм
Вес системного блока	6.58 кг
Напряжение питания	90–135 В, 60 Гц 198–264 В, 50–60 Гц
Потребляемая мощность	менее 100 Вт

Программирование контроллера прямого доступа к памяти

В.Плотников

Существует много особенностей программно-аппаратной организации ЭВМ IBM PC/AT, которые еще недостаточно изложены в отечественных публикациях. Это относится и к использованию средств прямого доступа к памяти (ПДП) указанных компьютеров.

Метод прямого доступа к памяти (Direct Memory Access — DMA) предназначен для организации обмена данными между ЭВМ и ее внешними устройствами (ВУ). Данный метод интересен тем, что пересылка данных происходит без участия процессора компьютера, и обмен с периферийным устройством единицей информации (байт или слово) осуществляется за один такт работы шины данных, а под управлением процессора — за два такта. Тем самым ПДП позволяет повысить производительность компьютера, так как появляется возможность параллельно с работой процессора выполнять операции ввода-вывода из ВУ. Причем наибольший эффект от такого режима работы достигается при выполнении процессором машинных инструкций с регистровыми операндами. Разработка программного обеспечения, основанного на поддержке DMA, расширяет функциональные возможности даже однозадачной операционной системы MS-DOS. Использование данного метода позволяет разрабатывать высокоэффективные драйверы устройств ввода-вывода информации, что необходимо при создании программных комплексов, обеспечивающих ввод и обработку информации в реальном масштабе времени. Применение для этих приложений традиционных методов ввода информации (обмен данными через порт) обеспечивает переработку данных, поступающих со скоростью до 100 Кбод. Аналогичные задачи, использующие ПДП, позволяют достичь скорости обмена до 1500 Кбод (для 8-битового канала DMA) или 3200 Кбод (для 16-битового канала DMA).

На компьютерах типа IBM PC/AT ПДП реализуется за счет наличия у них контроллера DMA, который выполнен в виде двух каскадно подключенных микросхем Intel 82C37A (или их модификаций). Такая аппаратная организация этого устройства позволяет иметь семь независимо программируемых каналов ПДП (каналы первого каскада 0-3 и каналы второго каскада 5-7), канал 4 используется для поддержки каскадного режима работы контроллера. Первый каскад контроллера обеспечивает обмен 8-битовыми данными (размер пересылаемого блока информации не должен превышать 64 Кбайтов), а второй каскад позволяет производить обмен 16-битовыми данными, при этом размер блока информации может быть до 128 Кбайтов. Необходимо помнить, что в IBM PC/AT часть каналов задействована для организации управления синхронной передачей данных (канал 1), накопителем на гибком магнитном диске (канал 2) и накопителем на жестком магнитном диске (канал 3). Поэтому для создания дополнительных ВУ могут быть использованы только каналы 0, 5, 6 и 7.

Программирование контроллера ПДП представляет собой процесс инициализации опреде-

ленных управляющих регистров (портов ввода-вывода компьютера). Реализация DMA доступа в программных разработках (в основном, в драйверах устройств) заключается в занесении информации в регистры:

терминального счетчика;
режима;
маски;
запроса;
базового адреса;
базовой страницы;
сброса.

Регистр терминального счетчика (размерность 16 битов) задает число передач в байтах (для 8-битового канала) или в словах (для 16-битового канала), которые должны быть выполнены при DMA доступе к устройству. Программирование этого регистра осуществляется помещением в него величины пересылаемого буфера, уменьшенной на единицу. После выполнения последней передачи значение счетчика станет 0FFFFh и произойдет аппаратная генерация сигнала конца пересылки данных TC, который может быть использован в схемах ВУ для возбуждения одной из линий аппаратных прерываний IRQ с целью выполнения определенных операций над заполненным информационным блоком или каналом DMA. Доступ к регистру терминального счетчика происходит как по чтению, так и по записи. Адреса портов ЭВМ для этого регистра приведены в табл. 1.

Таблица 1

Адреса расположения в памяти регистров терминального счетчика, базовых адреса и страницы

Номер канала DMA	Адреса портов расположения регистров		
	Терминального счетчика	Базового адреса	Базовой страницы
0	01h	00h	87h
1	03h	02h	83h
2	05h	04h	81h
3	07h	06h	82h
5	0C6h	0C4h	8Bh
6	0CAh	0C8h	89h
7	0CEh	0CCh	8Ah

Для инициализации DMA операции необходимо занести информацию, характеризующую используемый канал ПДП, в 8-битовый регистр режима. Он доступен прикладным программам только для записи и имеет адреса портов 0Bh (первый каскад) или 0D6h (второй каскад). На рис. 1 приведено назначение разрядов этого регистра.

Каждый канал DMA может осуществлять один из следующих типов обмена: проверка (тестирование канала), запись (данные поступают из ВУ в ОП) и чтение (перемещение информации из памяти в периферию).

Контроллер ПДП поддерживает четыре режима передачи данных: одиночный, блоком, по требованию и каскадный.

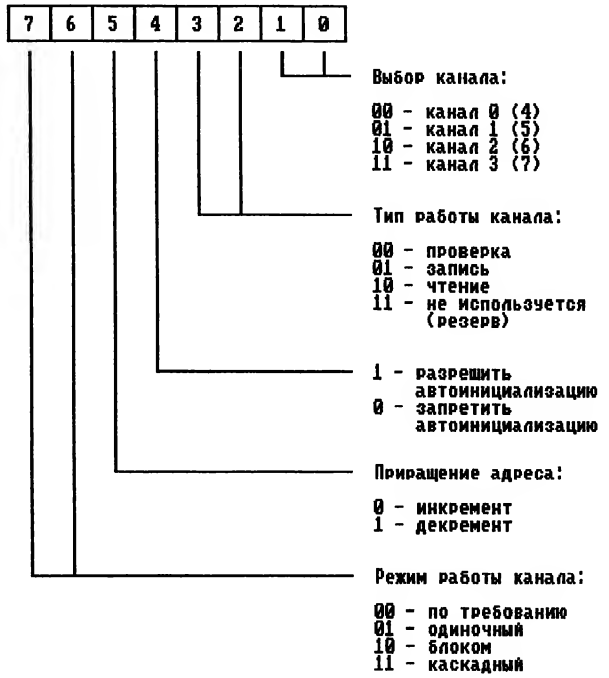


Рис. 1. Регистр режима

В режиме одиночного обмена устройство программируется, чтобы выполнить одну передачу (байт или слово), при этом терминальный счетчик уменьшается на 1 и контроллер на определенный промежуток времени освобождает шину данных для процессора или других каналов прямого доступа. Затем вновь возобновляется прием (передача) по данному каналу единицы информации, после чего снова канал блокируется. Так происходит до тех пор, пока не будет передан весь блок информации через данный канал.

Режим передачи по требованию предназначен для передачи данных через канал DMA до момента считывания (записи) всего блока информации, то есть появления сигнала TC на этом канале, или прекращения обмена данных с устройством (ВУ исчерпало объем своих данных). В любое время работы канала обмен информацией может быть приостановлен на некоторый промежуток времени для выполнения более приоритетного DMA обслуживания по другим каналам.

Работа канала при блочной передаче аналогична режиму передачи по требованию, за исключением того, что обмен информацией не может быть приостановлен более приоритетным запросом на шину данных до окончания блоковой передачи данных.

Каскадный режим используется для расширения возможностей архитектуры компьютера, чтобы логически соединить несколько контроллеров ПДП.

Задание приращения адреса позволяет установить флаг направления изменения адресов оперативной памяти (ОП), куда (откуда) происходит посылка (чтение) данных. Так, при сброшенном бите 5 регистра режима после посылки единицы данных (байт или слово) по каналу для организации следующей посылки текущий адрес ОП возрастает на величину

единицы информации, а при установленном бите адрес ОП уменьшается на эту же величину.

Если канал запрограммирован с автоинициализацией, то по окончании передачи данных (по сигналу TC) будут восстановлены первоначальные значения регистров терминального счетчика, базовых адреса и страницы, и пересылка информации вновь возобновится через данный канал.

Каждый каскад DMA имеет 8-битовый регистр маски, доступный прикладным программам только для записи (адрес порта вывода 1-го каскада — 0Ah, 2-го каскада — 0D4h). Он используется для управления программной блокировкой ПДП. Аппаратно канал блокируется установкой соответствующего бита этого регистра после окончания обмена информацией по сигналу TC (рис. 2).

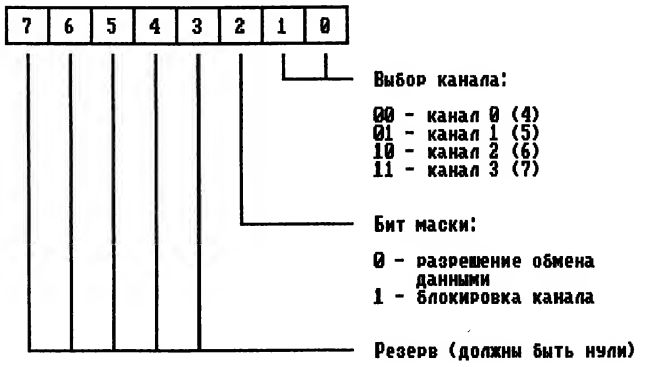


Рис. 2. Регистр маски

Регистр запроса используется программным обеспечением, чтобы инициировать запрос к каналу ПДП установкой бита 2 данного регистра. Для того чтобы выполнить этот запрос канал необходимо запрограммировать в режиме передачи блоком. Бит 2 может быть сброшен как программой (записью соответствующей маски), так и аппаратно (при выработке сигнала TC). Этот регистр доступен только для записи. При использовании первого каскада данные надо направлять в порт 09h, а второго — в порт 0D2h. Структура регистра запроса приведена на рис. 3.

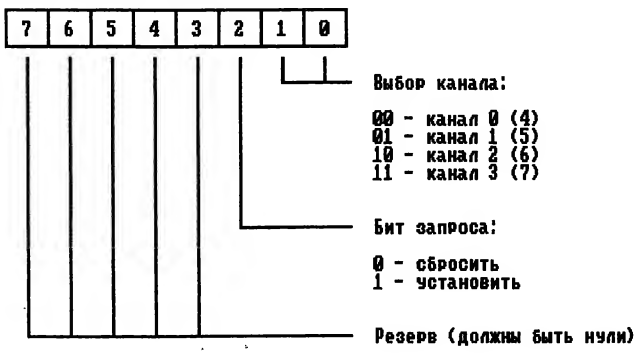


Рис. 3. Регистр запроса

При программной организации ПДП необходимо определить адресное пространство, с которым будет работать канал прямого доступа. Это осуществляется занесением информации в регистры базового адреса (разрядность 16 битов) и базовой страницы (разрядность 8 битов). Адреса портов компьютера для доступа к этим регистрам показаны в табл. 1. Прикладные программы могут не только заносить информацию в указанные регистры, но и в процессе работы читать из них значение текущего абсолютного адреса ОП (того участка памяти, куда в данный момент будет помещена единица информации). При инициализации DMA в данные регистры заносится абсолютный адрес начала буфера в ОП (если канал будет работать по инкременту адресов) или абсолютный адрес конца буфера в ОП (для декремента). Программирование этих портов ЭВМ для 8-битового и 16-битового каналов имеет существенное различие.

Базовый адрес для каналов первого каскада получается на основе вычисления из величин сегмента и смещения начала (конца) буфера 24-битового абсолютного адреса (при программировании в среде MS-DOS старшие 4 бита этой величины должны содержать 0). После чего младшие 16 битов этого значения помещаются в регистр базового адреса канала, а старшие 8 битов заносятся в регистр базовой страницы. Буфер, с которым будет работать 8-битовый канал ПДП, должен размещаться в пределах одного 64-килобайтового сегмента ОП (то есть в диапазоне абсолютных адресов: 0x0000h — 0xFFFFh).

Программирование значений этих регистров для 16-битовых каналов начинается также с вычисления 24-битового абсолютного адреса. Затем необходимо младшие 17 битов сдвинуть на 1 разряд вправо, обнулить 17-й бит адреса и полученное значение записать в регистры базового адреса и базовой страницы для обслуживаемого канала DMA. Буфер ОП может размещаться в любых адресах двух смежных сегментов памяти компьютера, но не пересекать их границы.

Регистр сброса (или триггера) предназначен для обеспечения 8-битового доступа по чтению и записи к 16-разрядным регистрам контроллера ПДП. После записи любого значения (контроллер не анализирует эту величину) в порт ЭВМ (для первого каскада по адресу 0Ch, а для второго — по адресу 0D8h) можно последовательно записать в любой 16-битовый регистр сначала младший, а затем старший байт.

Ознакомившись с правилами занесения управляющей информации в отдельные регистры контроллера, сформулируем обобщенный алгоритм программирования каналов ПДП (как 8-битового, так и 16-битового). Он заключается в выполнении следующих действий:

- занесение информации в регистр режима;
- запись в регистр сброса любого значения (это необходимо только в случае, если программное обеспечение не позволяет осуществить 16-битовый доступ к портам ввода-вывода компьютера);

– вычисление абсолютного адреса начала (конца) буфера ОП, к которому будет производиться прямой доступ, и запись полученного значения в регистры базовых адреса и страницы;
– загрузка регистра терминального счетчика величиной передаваемых единиц данных, уменьшенной на 1;
– установка программного запроса к DMA-обслуживанию записью в регистр запроса (если канал запрограммирован в режиме передачи блоком);
– сброс маски канала обнулением соответствующего бита регистра маски каскада.

Рассмотрим примеры практической реализации метода ПДП в прикладных программах.

Пусть имеется ВУ, принимающее блок информации по 8-битовому каналу ПДП с номером 0. Программа пользователя (листинг 1) должна считать данные из файла, инициализировать DMA для передачи их устройству. При неготовности устройства к приему данных программа пытается в течение 5 секунд продолжить посылку данных. Кроме того, примеры программ, работающих с 8-битовым каналом прямого доступа, можно найти в [1], [2].

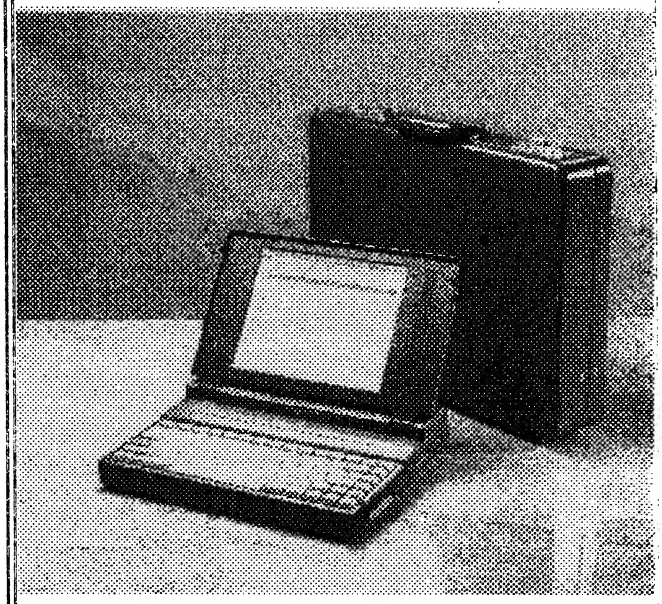
Обмен данными через 16-битовый канал ПДП иллюстрирует программа, приведенная в листинге 2. Она предназначена для получения блока данных от периферийного устройства и записи его на диск. Если от ВУ в течение 3 секунд не поступает информация, то на экран терминала выдается сообщение об этом.

Необходимо отметить, что описанные в данной статье принципы программирования ПДП характерны для работы контроллера в режиме совместимости адресов. При наличии у компьютера шины данных типа EISA программисту предоставляется возможность использовать каналы DMA для обработки данных, поступающих от 8-битовых, 16-битовых или 32-битовых источников. Организация 32-битового ПДП дает возможность пересылать информацию блоками до 16 Мбайтов (расширенный режим контроллера DMA). И в этом режиме прикладные программы работают по описанному в данной статье алгоритму — только в начале инициализации канала прямого доступа необходимо занести данные о емкости канала DMA (8 битов, 16 битов или 32 бита) в регистр расширенного режима. Более подробную информацию о работе контроллера в расширенном режиме можно найти только в документации фирмы Intel [3]. ■

ЛИТЕРАТУРА

1. Джордейн Р. Справочник программиста персональных компьютеров типа IBM PC, XT и AT/Пер. с англ. — М.: Финансы и статистика, 1991.
2. Фролов А.В., Фролов Г.В. Аппаратное обеспечение IBM PC. — М.: Диалог-МИФИ, Ч. 2. — 1992.
3. Peripheral components. — USA: Intel Corporation, 1991.

ВСЕГДА ЛУЧШИЙ!



Уважаемые господа!

PC Computing MVP MOST VALUABLE PRODUCT 1991 WINNER

EVEREX SYSTEMS, INC. продолжает специальную программу освоения российского рынка через своих дистрибьютеров и региональных дилеров.

Как всегда, мы предложим Вам все самое лучшее, от миниатюрного Notebook до супермощного мультипроцессорного сервера.

Посетив нас на выставке COMTEC'93, стенд № 274, Вы сможете оценить все преимущества EVEREX.

UNI Inc.
123376 Москва,
пер. П. Морозова, 5, стр. 1
Тел./факс (095) 205 47 49

CIA GmbH
117313 Москва,
ул. Кравченко, 4, корп. 2
Тел./факс (095) 138 11 41

intel inside

MICROSOFT® WINDOWS™ COMPATIBLE

EVEREX
EVER for EXcellence®

INTEL INSIDE™ & MICROSOFT® WINDOWS™ являются зарегистрированными торговыми марками.
EVER for EXcellence® является зарегистрированной торговой маркой фирмы EVEREX Systems, Inc.
© 1993, "Монитор"

Листинг 1. Программа вывода информации по 8–битовому каналу прямого доступа

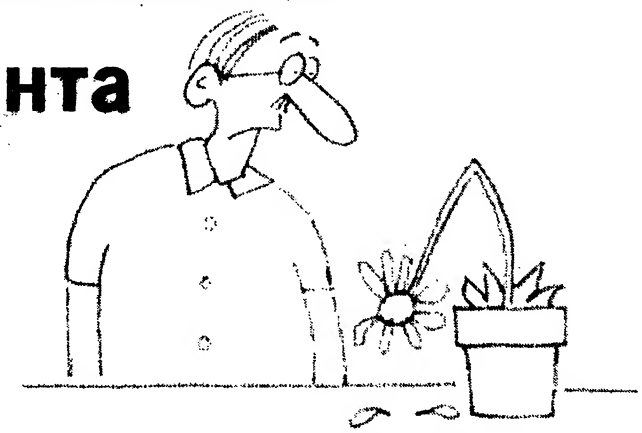
```
.286c
          title  dma8
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Программа получения блока данных из дискового
; файла текущего каталога с именем DMA8.DAT и
; вывод его на внешнее устройство ЭВМ
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
stack     segment   stack
          dw   120 dup(?)
stack     ends
dan       segment
fail      db   'dma8.dat',0      ; Имя файла для
                                 ; чтения данных
bufer     db   16 dup(?)         ; Буфер ОП для
                                 ; информации
soob      db   ' Неготовность устройства вывода',0Dh,0Ah,'$'
dan       ends
code      segment
          assume cs:code,ds:dan,ss:stack
dma8      proc  far
          mov  ax,dan
          mov  ds,ax

;
; Чтение информации из файла
;
          mov  dx,offset fail
          mov  ax,3D00h
          int  21h               ; Открываем файл
          mov  bx,ax
          mov  cx,16
          mov  dx,offset bufer
          mov  ah,3Fh
          int  21h               ; Читаем блок данных
                                 ; из файла
          mov  ah,3Eh
          int  21h               ; Закрываем файл
          mov  al,4              ; Маскируем канал 0
          out  0Ah,al            ; на период инициализации
;
; Инициализация 0-го канала DMA
;
          mov  al,00001000b      ; Маска режима:
                                 ;
                                 ; по требованию,
                                 ; нет автоинициализации,
                                 ; запись в устройство,
                                 ; канал 0
          out  0Bh,al            ; Записываем в регистр режима
          mov  si,ds
          mov  cx,offset bufer
          mov  ax,si
          shr  si,12
          shl  ax,4
          add  ax,cx             ; В si,ax получаем абсолютный
          adc  si,0              ; адрес начала буфера ОП
          out  0,ax              ; Записываем базовый адрес
          mov  ax,si
          out  87h,al            ; Записываем базовую страницу
          mov  ax,16             ; В ax - длина буфера в байтах
          dec  ax
          out  1,ax
          mov  al,0              ; Запускаем DMA операцию
          out  0Ah,al            ; размаскированием канала

;
; Посылаем данные во внешнее устройство
;
          mov  cx,90
dm1:
          hlt                    ; Ждем прерывания от таймера
          in   ax,1              ; Читаем регистр счетчика
          cmp  ax,0FFFFh
          jnz  dm2               ; Заданный буфер еще
                                 ; не полностью передан

          jmp  short dm3         ; Переходим на заключительные
                                 ; операции
dm2:
          loop dm1
          mov  al,4
          out  0Ah,al            ; Маскируем канал 0
          mov  dx,offset soob
          mov  ah,9
          int  21h               ; Выводим сообщение
                                 ; о неготовности устройства

;
; Заключительные операции
;
dm3:
          mov  ,4C00h
          int  21h
dma8      endp
code      ends
          end  dma8
```

Листинг 2. Программа ввода данных от внешнего устройства по 16–битовому каналу прямого доступа

```
.286c
          title  dma16
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Программа получения блока данных от внешнего
; устройства ЭВМ и записи его в дисковый файл
; текущего каталога с именем DMA8.DAT
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
stack     segment   stack
          dw   120 dup(?)
stack     ends
dan       segment
fail      db   'dma16.dat',0     ; Имя файла для
                                 ; размещения данных
          even
bufer     dw   16 dup(?)         ; Буфер ОП для приема
                                 ; информации
soob      db   ' Тайм-аут устройства ввода',0Dh,0Ah,'$'
dan       ends
code      segment
          assume    cs:code,ds:dan,ss:stack
dma16     proc  far
          mov  ax,dan
          mov  ds,ax
;
; Создание файла для записи информации
;
          mov  dx,offset fail
          mov  ah,3Ch
          int  21h
          mov  bx,ax
          mov  al,6              ; Маскируем канал 6
          out  0D4h,al           ; на период инициализации
;
; Инициализация 6-го канала DMA
;
          mov  al,01000110b      ; Маска режима:
                                 ;
                                 ; одиночная передача,
                                 ; нет автоинициализации,
                                 ; чтение из устройства,
                                 ; канал 6
          out  0D6h,al           ; Запись в регистр режима
          mov  si,ds
          mov  cx,offset bufer
          mov  ax,si
          shr  si,12
          shl  ax,4
          add  ax,cx             ; В si,ax получаем абсолютный
          adc  si,0              ; адрес начала буфера ОП
          test si,0001h
          jz   rea1
          and  si,0FFFEh
          stc
          jmp  short rea2
rea1:
          clc
rea2:
          rcr  ax,1              ; Формируем адрес
          out  0C8h,ax           ; Записываем базовый адрес
          mov  ax,si
          out  89h,al            ; Записываем базовую страницу
          mov  ax,16             ; В ax - длина буфера в словах
          dec  ax
          out  0CAh,ax
          mov  al,2              ; Запускаем DMA операцию
          out  0D4h,al           ; размаскированием канала
;
; Получаем данные от внешнего устройства
;
          mov  cx,48
rea3:
          hlt                    ; Ждем прерывания от таймера
          in   ax,0CAh           ; Читаем регистр счетчика
          cmp  ax,0FFFFh
          jnz  rea4              ; Заданный буфер еще
                                 ; не заполнен полностью
          mov  dx,offset bufer
          mov  cx,32
          mov  ah,40h
          int  21h               ; Записываем в файл буфер
          jmp  short rea5        ; Переходим на заключительные
                                 ; операции
rea4:
          loop rea3
          mov  al,6
          out  0D4h,al           ; Маскируем канал 6
          mov  dx,offset soob
          mov  ah,9
          int  21h               ; Выводим сообщение о тайм-ауте
;
; Заключительные операции
;
rea5:
          mov  ah,3Eh
          int  21h               ; Закрываем файл
          mov  ax,4C00h
          int  21h
dma16     endp
code      ends
          end  dma16
```

Заметки дилетанта

Ю. Косивцов



Изменение области данных BIOS

Идеальным редактором для написания и отладки программ на ассемблере, на мой взгляд, является MULTI-EDIT, а идеальным отладчиком TURBO DEBUGGER. Однако у MULTI-EDIT есть одно неприятное свойство: при компиляции программ внутри него изменяется область данных BIOS, относящаяся к видеоадаптеру. Например, он изменяет байт по адресу 0040:0084 (EGA — число символьных строк — 1), выставляя число символьных строк даже в случае отсутствия EGA адаптера. Конечно, это совершенно незаметно при работе и может иметь только "чисто научный интерес". Гораздо хуже то, что он изменяет слово по адресу 0040:004C (точнее, байт по адресу 0040:004D), которое содержит длину видеостраницы в байтах (нормально для текстового режима 1000h, устанавливает 5000h). Если после этого запустить TURBO DEBUGGER v.1.00, то он не будет работать (погасит экран и все). В качестве "противоядия" можно предложить небольшую программу, которая компилируется любым ассемблером, и создается COM файл длиной всего 8 байтов, который можно поместить в директорию с отладчиком и запускать отладчик через BAT файл (пример 1).

Изложенное относится только к TURBO DEBUGGER версии 1.00, версии 2.00 и 2.51 никак не реагируют на изменение данного слова в области данных BIOS. Однако, на мой взгляд, версия 1.00 является наиболее удачной для "медленных" машин (ИСКРА-1030, ROBOTRON CM-1910 и тому подобных), так как она имеет значительно меньший размер загрузочного файла и поэтому быстрее грузится, да и работает она быстрее. Так что, если не требуется отладка объектно-ориентированных программ, лучше всего пользоваться этой версией.

MULTI-EDIT не единственная популярная программа, которая изменяет область данных BIOS, относящуюся к видеоадаптерам. TECH-HELP v.3.2a by Dan Rollins выставляет байт по адресу 0040:0049 (текущий видеорежим), соответствующий видеорежиму, заложенному в конфигурации. Например, у Вас TECH-HELP установлен с использованием монохромного дисплея при адаптере CGA, тогда после окончания работы HELP байт по этому адресу будет равен 2, а если в конфигурации заложен цветной дисплей, то будет равен 3. Это вызывает неприятности при работе с продуктами фирмы BORLAND, например, TURBO DEBUGGER все будет выводить в черно-белом режиме.

Пример 1

```
CODE    SEGMENT  PARA    PUBLIC
        ASSUME   CS:CODE
        ORG      100H
START:
        MOV      DS,AX
        MOV      AL,10H
        MOV      BYTE PTR DS:[44DH],AL
        RETN
CODE    ENDS
        END      START                  ■
```

Отладка резидентных программ с помощью TURBO DEBUGGER

В TD версии 2.00 и выше имеется возможность отлаживать резидентные программы. Цитирую руководство пользователя: "Для отладки резидентных программ необходимо загрузить отлаживаемую программу, запустить ее на выполнение клавишей F9, затем найти точку входа в резидентную часть и поставить там точку прерывания клавишей F2. После этого необходимо оставить отладчик резидентным, для чего выбрать в меню FILE пункт RESIDENT. Отладчик останется резидентным в памяти и будет осуществлен временный выход в DOS, после чего необходимо выполнить действия по активизации резидентной программы. При проходе точки прерывания управление будет передано отладчику, после чего можно отлаживать резидентный участок программы."

Однако при попытке отладить таким способом обработчик INT 9, который наиболее часто используется для активизации резидентных программ, управление действительно будет передано отладчику, однако клавиатура будет заморожена из-за отсутствия сброса прерывания клавиатуры. Это наблюдалось как в версии 2.00, так и в версии 2.51. Чтобы исправить этот недостаток, можно воспользоваться следующей небольшой программой (пример 2). Действия при отладке в этом случае следующие: выполнить все действия, предусмотренные руководством пользователя, но перед тем как нажимать "горячую клавишу", загрузить эту программу. В этом случае при получении управления отладчиком клавиатура не будет заморожена.

Однако никогда не пытайтесь отладить описанным в инструкции способом обработчики прерываний 08 и 10h. Вы никогда не сможете выйти из отладчика в

DOS. Способ для их отладки следующий: в окне REGISTERS изменить на 3 слова регистр SP (отладчик автоматически поместит в них адрес возврата по IRET), затем заменить в том же окне регистр IP на точку входа в резидентную часть, войти в окно кода и, нажав CTRL–0, установить курсор на необходимое место. После этого программу можно отлаживать обычным путем в отладчике любой версии. Небольшое отличие заключается только в том, что версия 1.00 при старте помещает в стек лишнее слово, и указатель SP будет равен 0FFFCh вместо 0FFFEh для COM файлов.

Пример 2

```
; Данная программа позволяет отлаживать в
; Turbo Debugger v.2.0, v.2.51 обработчики прерывания 09.
; Она ассемблируется любым ассемблером, и создается
; COM файл. Действия по отладке резидентных программ
; описаны в документации на TD, за тем исключением,
; что после установки точки прерывания и оставления
; отладчика резидентным необходимо запустить данную
; программу, а уже затем нажимать "горячую клавишу".

TITLE   ДОПОЛНЕНИЕ_К_TD
CODE    SEGMENT  PARA PUBLIC  'CODE'
        ASSUME   CS:CODE, DS:NOTHING
        ORG      100H
START:
        JMP      SHORT   INSTALL

BREAKPOINT PROC  NEAR
        PUSH     AX          ; Сброс клавиатуры
        IN       AL,61H
        MOV      AH,AL
        OR       AL,80H
        OUT      61H,AL
        XCHG     AH,AL
        OUT      61H,AL
        MOV      AL,20H      ; Сигнал EOI на контроллер
                             ; прерываний
        OUT      20H,AL      ; Для владельцев
                             ; ROBOTRON CM-1910
                             ; необходимо вместо OUT 20h
                             ; написать OUT 0C0h
        POP      AX
        DB       0EAH        ; JMP FAR на обработчик
        IP_INT3  DW   0   ; INT 3 TD
        CS_INT3  DW   0
BREAKPOINT ENDP

INSTALL   PROC   NEAR
      MOV   ES,AX
      MOV   AX,WORD PTR ES:[OCH] ; Сохранение
      MOV   BX,WORD PTR ES:[0EH] ; адреса INT 3
      MOV   IP_INT3,AX           ; и установка его
      MOV   CS_INT3,BX           ; на свою
                                 ; процедуру
                                 ; обработки
      MOV   AX,CS
      CLI
      MOV   WORD PTR ES:[0CH],OFFSET BREAKPOINT
      MOV   WORD PTR ES:[0EH],AX
      STI
            MOV   AH,9
            MOV   DX,OFFSET MESSAGE
            INT   21H
            MOV   DX,OFFSET INSTALL
            INT   27H
MESSAGE     DB    'Дополнение к '
            DB    'TURBO DEBUGGER 2.0, '
            DB    'позволяет отлаживать INT 9'
            DB    0Dh,0Ah,'$'
INSTALL     ENDP
CODE        ENDS
            END   START
```

■

Для отладки резидентных программ, активизация которых осуществляется другими программами, лучше использовать другой отладчик (например, PERISCOPE, которым был отлажен драйвер, описанный в следующей части статьи).

Эмулятор мыши

Если эти строки попадутся на глаза Биллу Гейтсу, то скорее всего он посчитает, что они написаны ненормальным человеком. "Мышь" была придумана, чтобы облегчить и ускорить работу на компьютере, и человек, который пишет драйвер, эмулирующий ее работу, кажется, по меньшей мере, слегка свихнувшимся. Однако меня поймут те, кто весь свой рабочий день посвящает борьбе с такими машинами, как ИСКРА-1030, ROBOTRON CM-1910 и им подобными. Отсутствие последовательного интерфейса делает невозможным подключение стандартной "мыши", а "мышь" с параллельным интерфейсом пока является экзотикой. Один мой знакомый использовал для своих программ модуль TPMOUSE из библиотеки TURBO PROFESSIONAL и потом долго пытался отладить свою программу на ИСКРЕ. Сначала он просился к соседям, но дело дошло до того, что те стали закрывать перед ним дверь и делать вид, что никого нет. Специально для него был написан драйвер EMMOUSE, который позволяет осуществлять отладку программ, поддерживающих "мышь", на машинах, ее не имеющих.

Драйвер построен на основе прерываний от клавиатуры. Реализации могут быть различными. Для графических режимов я рекомендую использовать следующую схему: обработчик INT 9 выставляет определенные переменные на основании сигнала с клавиатуры, а обработчик INT 8 выполняет действия в соответствии с полученным сигналом. Для текстовых режимов обработчик INT 8 становится ненужным, "мышь" можно передвигать на одну позицию по одному прерыванию от клавиатуры. Кроме данных обработчиков, драйвер должен иметь обработчики INT 10h и INT 33h для отслеживания изменения видеорежима и обработки функций драйвера "мыши". К сожалению, я не могу привести листинг драйвера из-за слишком большого его размера, однако могу выслать свою реализацию всем желающим. Драйвер проверен на таких пакетах, как NORTON COMMANDER v.3.00, TURBO PASCAL v.6.00, TURBO DEBUGGER v.2.00, v.2.51, NU v.5.00, QUICK BASIC v.4.5. Реализованы также настройка на определенный тип клавиатуры и выбор управляющих клавиш. Возможно разделение серых и белых клавиш на 101-клавишной клавиатуре и клавиатуре CM-1910.

Несовместимость CM-1910 с IBM

Из-за несовместимости DMA ROBOTRON CM-1910 с IBM невозможно осуществление такой операции, как сброс в файл содержимого BIOS напрямую. Для осуществления этого необходимо сначала скопировать сегмент BIOS в оперативную память, а затем функцией 40h INT 21h сбросить в файл с установкой DS:DX на оперативную память. Если DS:DX установить на сегмент BIOS, то это вызовет зависание машины. ■

Ответ сердитому читателю

Письмо читателя

Господин Редактор!

Данное письмо — оценка одним из "программистов-практиков" характера публикаций Вашего журнала.

В целом, несмотря на рост цены, журнал представляет интерес для программиста. Но, к сожалению, его портят статьи достаточно общего содержания (про программирование вообще и хороший тон в программировании, в частности). В них приводятся советы, которым трудно возразить, так как они общеизвестны, и методы их выполнения, которые иногда лишь названы на словах, иногда намеренно усложнены.

Так, в Вашей заметке "Индикатор заражения программы компьютерным вирусом" (3.92) зачем-то требуется Турбо Ассемблер 2.0. По моим наблюдениям, программисты-практики ассемблером пользуются лишь тогда, когда ничто другое не подходит. Зачем задурять себе голову вопросами совмещения ассемблера с Паскалем, когда тот же результат можно получить гораздо короче (в смысле исходного текста) на Турбо Паскале? Да и понятнее, как отлаживать в оболочке:

```
program Test;
procedure Self_Test(Lold : LongInt);
var
    St      : string;
    Fin : file of byte;
    Lnew    : longint;
begin
    {$I-}
    Assign(Fin, ParamStr(0));
{ считываем имя исполняемой программы }
    Reset(Fin);
    Lnew := FileSize(Fin);
{ измеряем длину программы на диске }
    Close(Fin);
    {$I+}
    IF (LNew <> Lold) then begin
{ если разногласие — выдать сообщение }
        Writeln(' Длина файла изменена !');
        Writeln(' должно быть :', Lold:8);
        Writeln(' DOS сообщает:', Lnew:8)
    end;

BEGIN
    Self_Test(3);
{ в качестве аргумента следует указать реальный
размер }
END.
{ не правда ли, короче Вашего ассемблерного столбца? }
```

Сам способ защиты от вируса несколько ненадежен, так как множество современных вирусов не меняют длину заражаемой программы.

С большим удовольствием прочитал бы в Вашей статье, как загруженная в память программа может просчитать свою контрольную сумму (без работы с диском!).

Что касается ассемблера, то несомненно представляют интерес тонкости совмещения его с Паскалем, Си. Что в вашей заметке прозвучало лишь вскользь. Это тоже хотелось бы видеть на страницах "Монитора".

Тут я уже перешел к основной цели моего письма: Ваш журнал, по моему мнению, должен сделать акцент на публикации всяческих know-how практического программирования. Тонкости использования известных программ, замеченные в них ошибки (рубрика User => User).

Еще следует отметить, несмотря на конкуренцию с "Компьютер-Пресс", многие публикации по "железу" в Вашем журнале читаются с интересом. Хотелось, чтобы это было и дальше. А также про неизвестные свойства общеупотребимых адаптеров, устройств ввода-вывода, процессоров, и как все это можно пустить в дело?

Интересна и современная прикладная математика, прилагаемая к программированию. Например, листинг простой программы шифрования. (Не нужно качества на полгода взламывания Cray-2, многих практиков устроит и час!). Если не ошибаюсь, это даже в каком-то ГОСТе есть.

Или — простейшая реализация цифровой подписи. Или — листинг системы ввода-вывода большого числа данных с диска со сжатием их. Обзор современных алгоритмов сжатия, с тем, чтобы какой-нибудь из них читатель мог реализовать у себя на машине.

Конечно, если пойти в библиотеку и там это все найти (если библиотека хорошая), можно за недельку проштудировать книжку и самому все это воплотить в виде соответствующего модуля или объекта. Но, по моим наблюдениям, у подавляющего числа практиков (не у высоких профессионалов, создающих классный продукт с огромным тиражом) до этого просто руки не доходят.

Я бы хотел иметь подобные листинги на своей книжной полке в Вашем журнале. Да и листинг совсем не обязателен, главное, чтобы алгоритм был изложен так, что только по Вашей публикации можно было бы довести дело до кода.

С надеждой стать вашим постоянным читателем

Е.Гавриш ■

Ответ редактора

Ну что же, давайте поговорим об "ассемблерных извращениях". Я позволю себе не согласиться с тезисом, что ассемблером пользуются только тогда, когда ничего больше не подходит. Основная цель разработки ассемблерных процедур для программ на языках высокого уровня — это скорость исполнения, объем кода и его надежность [1]. В нашем случае основное внимание уделялось минимизации кода и повышению его надежности. Но что понимается под надежностью? При решении вирусных проблем это, прежде всего, уверенность в том, что процедура выполняет только те операции, которые от нее требуются, и более никаких. Любая избыточность создает дополнительную лазейку для вирусов и снижает эффективность разработанной защиты.

Теперь давайте рассмотрим, во что выливается Ваше предложение. В примере 1 приведен результат дисассемблирования (с помощью

Пример 1

```
TEST.SELF_TEST:  begin
      push   bp
      mov    bp,sp
      mov    ax,0284
      call 5092:027C
      subsp,0284
TEST.17:     Assign(Fin, ParamStr(0));
      lea di,[bp-0180]
      push   ss
      push   di
      lea di,[bp-0284]
      push   ss
      push   di
      push   0000
      call 5092:0B4D
      call 5092:0910
TEST.19:     Reset(Fin);
      lea di,[bp-0180]
      push   ss
      push   di
      push   0001
      call 5092:093E
TEST.21:     Lnew := FileSize(Fin);
      lea di,[bp-0180]
      push   ss
      push   di
      call 5092:0AD8
      mov    [bp-0184],ax
      mov    [bp-0182],dx
TEST.23:     Close(Fin);
      lea di,[bp-0180]
      push   ss
      push   di
      call 5092:09BF
```

Turbo Debugger 3.0) предложенной Вами процедуры Self_Test. Как видно из рисунка, уже на этом этапе появляется пять (!) дальних вызовов подпрограмм. Попробуем проследить только одну из них — подпрограмму вычисления длины файла (пример 2).

Пример 2

```
5092:0AD8
  call    0B08   ; вызов процедуры определения длины файла
  jb      0AF0
  mov     ax,cx  ; перенос результатов
  mov     dx,bx
  mov     cx,es:[di+04]
  cmp     cx,0001
  je      0AF0
  xor     bx,bx
  push    cs
  call    0769
  retf    0004
  call    0B08
  jb      0B03
  sub     ax,cx
  sbb     dx,bx
  jb      0B03
  mov     al,01
  retf    0004
```

Однако и на этом этапе мы не получаем полной ясности. Появляется еще один вызов, на этот раз ближний. И только спустившись еще глубже (пример 3), мы обнаруживаем, как на самом деле Turbo Pascal определяет длину файла. А вот теперь я задам пару ри-

Пример 3

```
cs:0B08
  mov     bx,sp        ; какая-то проверка записи файла
  les     di,ss:[bx+06]  ;
  cmp     es:word ptr [di+02],D7B3
  jne     0B41
  xor     cx,cx
  xor     dx,dx
  mov     bx,es:[di]   ; загрузить в BX номер канала (handle)
  mov     ax,4201      ; считать текущее положение файла
  int     21
  push    dx           ; сохранить смещенин от начала файла
  push    ax
  xor     cx,cx
  xor     dx,dx
  mov     bx,es:[di]   ; снова загрузить в BX номер канала (handle)
  mov     ax,4202      ; установка указателя в конец файла
  int     21
  pop     bx
  pop     cx
  push    dx           ; сохранение длины файла в стеке
  push    ax
  mov     dx,bx
  mov     bx,es:[di]   ; еще раз загрузить в BX номер канала (handle)
  mov     ax,4200      ; установка указателя на прежнее место
  int     21
  pop     cx       ; BX:CX - длина файла
  pop     bx
  ret
```

торических вопросов. Нужно ли таскать за собой из программы в программу весь этот явно избыточный код? Способствует ли повышению надежности разрабатываемых программ включение "лишних", с точки зрения решаемых задач, команд? Необходимо ли тратить оперативную память (которой всегда не хватает) на поддержку совершенно лишних структур данных (посмотрите, например, что представляет собой тип данных File в понимании Turbo Pascal [2])?

```
type
{ типизированные и нетипизированные файлы }
FileRec = record
    Handle : Word;
    Mode   : Word;
    RecSize    : Word;
    Private : array [1..6] of Byte;
    UserData: array [1..16] of Byte;
    Name   : array [0..79] of Char;
end;
```

Что же касается отладки процедуры на ассемблере, то указанный фрагмент был запущен, что называется, "с листа". Кроме того, хочу напомнить, что версия Turbo Pascal 6.0 имеет встроенный ассемблер с полной поддержкой отладки ассемблерных текстов в IDE.

И, наконец, о надежности критерия. Да, конечно, некоторые вирусы не изменяют или маскируют изменение длины поражаемого файла. Но подавляющее большинство устроено куда проще (Е.Касперский называет их "студенческими" вирусами). А для более изощренных, тут Вы, пожалуй, правы, целесообразно использовать подсчет контрольной суммы программы. Но об этом мы поговорим отдельно.

В.Водолазкий

Литература

1. **В.Водолазкий, В.Головин, Е.Короткова. Паскаль + Ассемблер... или как использовать "мышь" в графических режимах//Монитор. — 1992.- № 5.**
2. **Turbo Pascal 6.0. Programmers guide. — Borland International, 1991.**

Уважаемые читатели!

Мы предлагаем Вашему вниманию

компьютерный "ларек".

Мы преследуем две основные цели:

— дать возможность отечественным разработчикам инструментальных средств выйти на российский рынок программного обеспечения;

— обеспечить как можно более широкий круг читателей недорогими и оригинальными средствами разработки программного обеспечения.

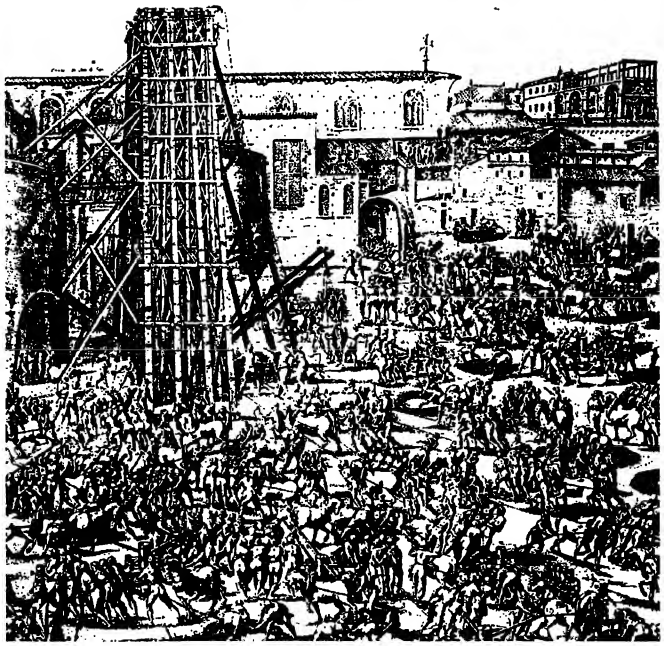
Код	Продукт	Краткая характеристика	Объем поставки	Цена, $	Демо	Установщик	Требования к оборудованию	Сопровождение пользователей	Число продаж	Скидка по upgrade	Скидки в оплате	Примечания
SL-01	Turbo Games	Объектно-ориентированная библиотека для Turbo Pascal 6.0. Спрайты, звуковые эффекты, трехмерная графика, стереоэффекты. Доступ к ZIP файлам.	2 дискеты + документация	12 для индивидуальных покупателей, 25 для организаций	+	+	EGA адаптер, IBM PC/AT	+		+	+	Готовится версия для VGA
SL-02	Data Vision 1.5	Объектно-ориентированная библиотека для Turbo Pascal 6.0. Создание и управление реляционными базами данных. Поддерживаются DBF и DB форматы. Поддержка электронных таблиц.	2 дискеты + документация	12	+	—	IBM PC/XT	+	37	+	—	Версия 2.0 (осень 93 года) будет поддерживать файлы СУБД Clanor и содержать генератор отчетов
IS-01	HyperRef	Инструментальное средство для создания гипертекстовых систем. Поддержка текстовой информации, изображений и исполняемых программ.	2 дискеты по 1,2 Мб	35	+	—	IBM PC/AT, EGA/VGA, 3 Мб свободного пространства на HDD, 350 Кб ОЗУ- просмотр документа, 450 Кб – редактирование документа.					
CS-01	QuickTools V1.0	Средство разработки диалога в Paradox.	1 дискета 1,2 Мб или 1,44 Мб	10	+	+	IBM PC/AT, EGA/VGA, 3 Мб на HDD, 1-2 Мб ОЗУ.		+			
UT-01	ArcView 6.1	Просмотр и редактирование заголовков архивных файлов с возможностью подключения собственных типов архивов.	1 дискета 360 Кб	1	—	+	IBM PC/AT, Hercules, EGA/VGA	+				
SE-01	Crypter 2.0	Система шифрования файлов. Поддерживаются алгоритмы DES, ГОСТ	1 дискета 360 Кб	12	—	+	IBM PC/AT, Hercules, EGA/VGA	+				
IS-02	OKO 1.0	Инструментальное средство для разработки приложений, интерфейс сходен с интерфейсом электронной таблицы. Позволяет быстро переносить документы с PC и объединять их в единую систему обработки данных. Реализовано как верхняя оболочка сетевой СУБД dbVISTA.	4 дискеты по 360 Кб или 1 дискета 1,2 Мб Документация 2 файла на установочной дискете.	18	+	+	IBM PC/XT, EGA/CGA/VGA, 1 Мб HDD, 640 Кб ОЗУ	+		+ (18 $ за 1-ю установку и 3 за каждую последующую)	+ (20% для новых версий для пользователей OKO)	
SC-01	АвтоГРАФ	Пакет для создания на экране графиков с результатами обработки числовых данных и печати их на 9- и 24-игольчатых принтерах. Поддержка форматов ЕСКД.	1 дискета 360 Кб	10	+	+	IBM PC/XT, EGA/VGA, 500 Кб на HDD, ОЗУ – 200 Кб	+	70	+	+ для учебных заведений	Готовится версия 3.0

Если Вас заинтересовали приведенные здесь продукты или Вы хотите получить дополнительную информацию, звоните, пишите, приезжайте.

Оплата принимается только в рублях по текущему курсу.

Адрес для писем: 117419 Москва, а/я 765

Тел.: (095) 238-99-33. Факс: (095) 238-32-78



Игры с несколькими участниками

Часть I

Те, кто постоянно читают журнал "МОНИТОР", конечно же, обратили внимание на регулярные отчеты об исследованиях игровых программ, публикуемые малым научно-техническим предприятием "ИНФОРКОМ". У этих исследований есть вполне конкретная цель. Поскольку "ИНФОРКОМ" — одна из немногих в стране компаний по программированию, занимающаяся вопросами обучающего программного обеспечения, то и основное направление этих исследований связано с изучением тех концепций и закономерностей игрового программного обеспечения, которые могут быть положены в основу обучающих систем. А таких концепций и методических идей, как показывает наш четырехлетний опыт работы в данном направлении, очень и очень много.

Сегодня мы хотели бы обратиться к читателям журнала, ценящим и знающим игровые программы, с просьбой внести и свой вклад в это исследование. Дело в том, что "ИНФОРКОМ" проводит постоянный опрос общественного мнения с целью выяснения вкусов и пристрастий различных слоев пользователей. Мы просим Вас назвать пять-шесть Ваших любимых игровых программ и, по возможности, указать год выпуска и фирму-производителя. Еще большую услугу Вы окажете нам, если сочтете возможным при этом указать свой возраст и род занятий. Мы очень надеемся, что Вас не затруднит направить эту информацию по нашему адресу:

121019 Москва, Г-19, а/я 16, "ИНФОРКОМ"

Если Вы не интересуетесь игровыми программами, но у Вас есть дети 7-9 лет, которые играют в игры на компьютере, то нам очень интересно было бы знать названия этих игр и Ваше впечатление о том, как проходят сеансы работы детей с компьютером. Наши исследования показывают, что есть очень и очень большая проблема с подбором игротеки именно для этого возраста, и Ваше мнение для нас чрезвычайно важно. Мы же, в свою очередь, готовы познакомить заинтересованные круги с результатами подобных исследований.

А сейчас, продолжая наш разговор об игровых программах, мы вернемся к тому направлению, о котором упомянули в прошлых выпусках, — Role Playing Games. Как следует из названия, это игры для нескольких участников, каждый из которых играет отведенную ему роль.

История жанра

Игры Role Playing возникли еще до появления первых персональных компьютеров. Их историю можно проследить от настольных военно-стратегических игр (WARGAMES), в которых играющие управляют крупными войсковыми соединениями и объединениями. Осознание того очевидного факта, что каждый человек представляет собой яркую индивидуальность и неповторимую ценность, и привело к созданию игр, в которых вместо армий и флотов играющие стали управлять несколькими героями с различным игровым

амплуа. Широкое распространение RP игры получили как игры по переписке — PBM Games (Play by Mail). Первая коммерческая компания-организатор игры по переписке появилась в штате Висконсин, США в середине 70-х годов. Так было положено начало движению, в которое оказались вовлечены десятки миллионов долларов и сотни тысяч людей.

Появление первых персональных компьютеров привело к попыткам адаптации жанра в рамках компьютерной игры. Самой первой игрой, родоначальницей жанра, считают написанную в 1980 году игру "Dungeons & Dragons". Это еще не была в полном смысле игра Role Playing, потому что главный герой в игре был только один, но она надолго предопределила закономерности развития жанра. С тех пор в терминологии игровых программ принято обозначать игры, в которых основная цель состоит в том, чтобы провести героя по неисследованным лабиринтам, как D&D.

Настоящие компьютерные Role Playing игры появились несколько позже, когда размер оперативной памяти машин стал достаточным для размещения в ней информации о нескольких героях.

Дальнейшее развитие жанра продолжалось с появлением компьютерных сетей и возвращением играм первоначального статуса многопользовательских — MUG (Multiuser Games). Первый образец такой игры появился в Эссекском Университете (Англия) в 1984 году. Такие многопользовательские игры доступны на Западе всем, у кого есть компьютер с последовательным портом RS232 и модем. Уже в первый год своего существования многопользовательская игра объединила до 1200 участников, и к настоящему времени существуют тысячи подобных проектов, подключиться к которым может каждый, кто внесет вступительный взнос и будет платить абонементную плату. Тематический спектр игр чрезвычайно широк. Вы можете попробовать свои силы в самых разных ролях: от средневековой колдуньи до капитана звездолета, терпящего бедствие во враждебной планетной системе. Впрочем, MU игры для нас пока остаются делом будущего и сейчас мы сосредоточимся на классических компьютерных образцах RP Games и постараемся подвести читателя к тому, какие идеи и концепции, положенные в основу жанра, смогут быть использованы при создании обучающего программного обеспечения.

Особенности жанра

Как и в программах жанра ADVENTURE, в этих играх существует более или менее ярко выраженный сюжет и перед участником стоит цель, которой ему надо достичь, хотя выяснение того, что это за цель, может быть отдельной самостоятельной задачей (серия "Ultima" фирмы ORIGIN).

К характерным особенностям игр этого класса надо отнести, во-первых, размер игрового пространства, заметно превосходящий адвентюрные игры, — несколько тысяч локаций. Это приводит к тому, что разнообразить геометрию игры становится достаточно трудно, и игровое пространство сводится к координатной сетке или к набору координатных сеток. Текущее местоположение персонажей изображается в схеме "вид сверху" или "слепой лабиринт". В некоторых играх используется их комбинация. Иногда пользователю доступна карта или ее фрагмент, иногда же картографирование ведется автоматически.

Второй особенностью является то, что каждый персонаж (а в этих играх, как правило, действует не один герой, а целая команда) обладает набором свойств, каждое из которых определяется динамически изменяющимися параметрами. Значения этих параметров влияют на способности персонажа (в основном на боевые). Некоторые из этих параметров могут изменяться автоматически по мере увеличения боевого опыта, а для изменения других могут потребоваться особые обстоятельства.

Третьей особенностью игр этого жанра является то, что встречающиеся в игре логические головоломки обычно довольно просты и сводятся, как правило, к сбору информации и поиску путей доступа в новую часть игрового пространства, то есть задача сводится к поиску ключей или объектов, выполняющих аналогичные функции. Логический же акцент перенесен на выбор правильной тактики, а иногда и стратегии взаимодействия с враждебными существами и их объединениями. В связи с этим основная масса объектов, встречающихся в таких играх, имеет прикладное и боевое значение: оружие, доспехи, магические свитки с заклинаниями, провизия и тому подобное. Нередко важную роль играют деньги, которые обычно бывают военной добычей. Во многих играх деньги являются средством конверсии одних видов оснащения в другие и наоборот. В этом случае в игре дополнительно появляется "деловой" аспект.

Современные игры RPG имеют, как правило, высококачественную графику. Конфигурация игрового поля обычно задана изначально, а размещение объектов по локациям однозначно определено. Случайность и разнообразие, вводимые в поведение или размещение враждебных сил, не настолько велики, чтобы вызвать желание повторно сыграть в уже пройденную игру, зато обширный объем игрового пространства достаточен, чтобы удерживать внимание пользователя в течение длительного времени.

Персонажи

В большинстве игр такого рода по игровому пространству путешествует отряд героев, действующих сообща. Обычно в начале игры задается ядро отряда (около двух третей максимального состава), а возможность дальнейшего его пополнения возникает по ходу игры. В игре же "King's Bounty" фирмы NEW WORLD COMPUTING играющий выступает в роли командующего наемной армией, размер которой зависит от личного авторитета героя и его финансовых возможностей.

В тех играх, где боевой отряд состоит из отдельных персонажей, в начале игры обычно есть возможность влиять на их исходные характеристики. Эти характеристики либо разыгрываются при помощи генератора случайных чисел, либо задаются в диалоговом режиме, при этом используются ответы пользователя на вопросы, которые никакого отношения к настраиваемым параметрам могут и не иметь. Это одно из проявлений концепции искусственного интеллекта (ИИ), очень часто эксплуатируемой в играх RPG. Во многих играх предусматривается и возможность ручного задания исходных характеристик персонажей, например, в сериале "Eye of Beholder" фирмы WESTWOOD ASSOCIATES.

Интересно, что в различных играх используется одна и та же система, позаимствованная из игр класса DUNGEONS & DRAGONS. В этой системе параметрами героя являются: сила, ловкость, телосложение, интеллект, мудрость и обаяние. Эти параметры могут меняться в широких диапазонах, например, от 3 баллов до 18. Первые три параметра влияют в основном на боевое искусство героя, а остальные — на его магические способности и способность общения с другими персонажами. Защищенность героя надетыми на него доспехами отсчитывается от величины 10 (полное отсутствие доспехов) в сторону уменьшения и может выражаться и отрицательным числом.

Кроме того, в некоторых играх (в том числе у разных фирм-производителей) стандартным стал выбор персонажа по мотивам шотландских, кельтских, скандинавских и германских мифов (люди, эльфы, гномы, гоблины и многие другие). При отсутствии фирменной документации по внешним проявлениям создается впечатление, что этот параметр не оказывает существенного влияния, хотя некоторая разница иногда бывает заметна. Стоит упомянуть и возможности задания пола персонажей и даже выбора его портрета, которые, впрочем, также не оказывают особого влияния на ход игры.

Помимо этого, в большинстве игр у каждого героя бывает еще и амплуа, определяющее стратегию и тактику его использования. Так, различные типы воинов (Fighter, Paladin, Ranger, Knight) в разных играх имеют обычно наиболее универсальные навыки владения различными типами оружия. В поединках они участвуют прямым и непосредственным образом.

Священник (Cleric, Priest), как правило, ограничен в выборе ручного оружия, но имеет широкий спектр чудодейственных заклинаний, направленных, в основном, на лечебные и защитные цели. Такие персонажи могут в промежутках между боями восстанавливать силы и здоровье других персонажей.

Чародей (Mage, Sorcerer, Magic User) также имеет в своем распоряжении набор заклинаний, но этот набор обычно практически не пересекается с возможностями священника и имеет ярко выраженную боевую направленность.

Разбойник (Thief, Robber) так назван в силу исторических традиций и понимать это амплуа следует скорее как разведчик. Обычно он имеет достаточный набор возможностей по ручному оружию, а его особенностью является умение открывать без ключей запертые двери, сундуки, быстро и незаметно перемещаться.

Реальные возможности персонажей увеличиваются по мере накопления боевого опыта. В большинстве игр существует параметр опыта (Experience), постоянно растущий по мере побед над противниками и в ходе

Рецензия на книгу В.Ю. Романова "Популярные форматы файлов для хранения графических изображений на IBM PC"

Появления подобной книжки [1] следовало ожидать еще год или два назад. Но, несмотря на такую задержку, СП "Унитех" сумело оказаться в первых рядах просветителей, предоставив широкому кругу программистской общественности информацию о форматах графических файлов, которую можно было получить ранее только через доступ к BBS.

В книге рассматриваются четыре наиболее популярных формата: PCX, GIF, TIFF, PIC (Lotus). Информация о каждом из них рассматривается независимо, что приводит к неизбежным повторам, в частности, при рассмотрении алгоритма упаковки данных LZW. Но это не является недостатком, скорее, наоборот. Для каждого из описываемых форматов автор приводит полные исходные тексты программ на Си, осуществляющих сохранение изображения на экране в файле соответствующего формата, а также визуализации изображения, хранящегося в файле, на экране.

Да, кстати об экране. Программы, приведенные в книге, ориентированы на использование EGA адаптера. Для того чтобы читателю не пришлось искать руководство по программированию EGA, автор включил в содержание краткое описание используемых режимов работы адаптера и тексты соответствующих процедур на Си. Так что книга имеет законченный вид и может читаться (и применяться) без использования дополнительных источников.

Необходимо отметить, что В.Романов не ограничился переводом и компиляцией текстовых файлов с описаниями графических форматов, доступными из ряда BBS, в частности, московских. Приятно радует глаз подробное описание алгоритмов архивации графических данных : RLE для PCX, LZW для GIF, битовая упаковка, LZW и код Хаффмена для TIFF. И, естественно, для каждого из этих алгоритмов написана соответствующая процедура.

В целом издание производит благоприятное впечатление. Но не могу удержаться, чтобы не добавить от себя ложку дегтя. Нет, моя ложка относится не к содержанию, а к оформлению. К сожалению, приведенные в книге таблицы со структурами данных оформлены не слишком наглядно. В качестве примера рассмотрим описание байта "Разное" из заголовка файла формата GIF.

исследования игрового пространства. Приобретая новый опыт, персонаж получает возможность достичь нового уровня (Level), что, как правило, повышает некоторые из его характеристик. Магические способности, а в некоторых случаях и набор доступных заклинаний обычно также при этом возрастают. Переход на новый уровень совершается либо автоматически, либо при посещении какой-то особой локации в игровом пространстве (например, тренировочного зала или чего-либо подобного). Такой "рост" персонажа совершенно необходим для успешного прохождения игры. В некоторых играх повышение уровня возможно всего лишь несколько раз за игру, в других же персонаж, начавший с первого уровня, должен приблизиться к сотому, чтобы получить шансы на победу в бою с самыми сильными из противников.

Снаряжение и другие объекты

Количество предметов, одновременно находящихся при персонаже, в таких играх обычно довольно велико, но все же ограничено по количеству или по весу. Надетое на персонаж снаряжение может входить, а может и не входить в эти ограничения. Некоторые специальные предметы иногда засчитываются особым образом. Деньги, например, либо засчитываются отдельно, либо входят в общий вес, либо сумма наличных денег может быть неограниченной. Лишнее снаряжение иногда приходится выбрасывать, а в некоторых случаях его можно конвертировать в деньги. Снаряжение может либо приобретаться, либо достается в качестве боевого трофея, либо разыскивается при исследовании игрового пространства. В некоторых играх можно рекомендовать сохранять с собой все, что удалось найти. Например, в игре "Eye of Beholder" лабиринт устроен так, что для проникновения в новые области приходится использовать предметы самым неожиданным образом. В некоторых играх стоит расставаться со всем, что не нужно, как можно быстрее (см. "Might and Magic" фирмы NEW WORLD COMPUTING), потому что при отсутствии свободного места лишнее снаряжение будет просто потеряно. Интересна особенность многопользовательских игр, в которых играющие расстаются с излишним снаряжением, не выбрасывая, а уничтожая его. Это делается для того чтобы вероятный противник не смог впоследствии им воспользоваться. Характерным для таких игр стал прием "утопить сокровище в болоте".

Оружие и амуниция, а иногда и магические предметы должны быть подготовлены для использования (надеты, взяты в руки и так далее). Как правило, стоит вооружать и снаряжать персонажей лучшим из того, что имеется в наличии. Тем не менее, все же следует иметь в виду, что поединки с некоторыми из противников могут привести к уничтожению или существенному ухудшению возможностей некоторых предметов.

* * *

На этом мы временно прервем наш рассказ о Role Playing играх и вернемся к ним в следующем выпуске, где расскажем о характере исследования игрового пространства, об особенностях боевого поведения персонажей, об использовании магии, о тактике и стратегии играющего и о возможности использования базовых концепций жанра в развитии средств обучающего программирования. ■

"Байт Разное содержит следующую информацию:

* M (1 бит) Признак наличия глобальной карты цвета.
* R (3 бита) Разрешение цвета. Сколько битов выделяется на значение цвета. Если эти три разряда равны 0, то на цвет выделяется один бит и так далее.
* X (1 бит) Всегда 0.
* C (3 бита) Максимальное число цветов в изображении. Числа от 0 до 7 в этих разрядах представляют от 1 до 8 битов. Следовательно, цветов может быть от 2 (черно-белое изображение) до 256."

И в каком же порядке перечислены описанные здесь значения битовых полей? Начиная со старшего бита в байте или с младшего? Готовы ли Вы ответить на этот вопрос, не заглядывая в листинг программы? А ведь насколько удобнее было бы разобраться в формате, если бы автор добавил графическую интерпретацию структуры байта "Разное", как это и сделала фирма Compuserve в описании своего стандарта GIF.

То же самое можно сказать и о многобайтовых структурах данных, в частности, обо всех заголовках файлов, признаках TIFF формата и так далее.

(Стоило ли выносить в выходные данные тот факт, что оригинал-макет подготовлен на компьютерах Macintosh, если даже реализация подобных иллюстраций встретила, вероятно, некие непредвиденные трудности? Наш журнал подготавливается на IBM PC/AT с помощью пакета Ventura Publisher фирмы Xerox, и его оказывается вполне достаточно. Нужно ли гробить миллионы на Apple Macintosh? — примечание верстальщика).

Еще одним недостатком, который не так заметен на первый взгляд, является чрезмерная лаконичность при описании истории создания и области применения рассматриваемых форматов. Кроме того, совершенно выпущен из виду вопрос о совместимости, более того, о существовании различных версий графических форматов. Так что содержащуюся в ней информацию можно оценить как достаточную для практического применения некоторого (хотя и достаточно широкого) подмножества графических файлов, но не как исчерпывающую.

В целом же издание, как уже указывалось, производит благоприятное впечатление и может использоваться в качестве справочного руководства. Красивых Вам картинок!

В.Водолазкий ■

Литература

1. В.Ю.Романов. "Популярные форматы файлов для хранения графических изображений на IBM PC". — М.: Унитех, 1992.

Caviar приходит в Россию

О фирме Western Digital из США в нашей стране специалисты в области вычислительной техники и пользователи узнали относительно недавно. Но с момента своего основания в 1970 году фирме удалось занять влиятельное положение на мировом рынке производителей полупроводников, интегральных микросхем, а также расширить свою деятельность в сфере производства жестких магнитных дисков (винчестеров) и плат графических ускорителей. Western Digital сегодня — это транснациональная компания, имеющая свои филиалы в восьми штатах США, Франции, Германии, Гонконге, Японии, Корее, Великобритании и на Тайване.

С 1988 года Western Digital становится одним из ведущих производителей винчестеров со встроенным контроллером (intelligent drive electronics — IDE). А быть одним из первых — это не только прибыли и лавровые венки. Это еще и ответственность. Ведущий производитель, как это убедительно демонстрирует пример фирмы IBM, фактически предопределяет направление развития отрасли на несколько лет, если не десятилетий, вперед.

Western Digital гордится своим комплексным подходом при разработке новых моделей. Фирма ведет ряд научно-исследовательских и опытно-конструкторских работ по улучшению существующего ряда выпускаемых изделий и по разработке новых продуктов и технологических процессов. Поскольку Western Digital владеет динамически развивающейся долей рынка, отставание в которой может привести к полной потере конкурентоспособности изделий, необходимость постоянно находиться на острие научно-технического прогресса становится жизненно важной. По заявлению представителей фирмы, положение компании на рынке в значительной степени определяется успехами ее разработчиков. Достаточно привести только одну цифру: 49% общего объема прибыли в четвертом квартале 1992 года было получено от продажи продуктов, разработанных позже 4 квартала 1991 года.

Куда же направлена исследовательская деятельность Western Digital? Прежде всего, компания ориентируется на увеличение емкости винчестеров и проводит соответствующую маркетинговую политику. Уже сейчас винчестеры емкостью 120 и 170 мегабайтов стоят примерно одинаково. Western Digital считает, что наиболее успешно в 1993 году будет продаваться модель Caviar 340, а модели емкостью 425 Мб, презентация которых состоялась на выставке Cebit-93, во втором квартале 1993 года запущены в серийное производство (рис. 1). Western Digital уже сняла с производства 3.5" модель винчестера емкостью 40 Мб и планирует в 1993 году прекратить выпуск винчестеров емкостью 80 и 120 Мб. Кроме того, компания прилагает ряд усилий по замене популярных моделей более новыми и более надежными. Так, модель WDAC2170 (с двумя дисками, рис. 2), появившаяся во втором квартале 1992 года, уже в 1993 году заменяется на WDAC1170, представляющую собой как бы "половинку" двухдискового накопителя WDAC2340 емкостью 340 Мб.

Ускоренное внедрение в практику передовых разработок (новая модель появляется почти в каждом квартале) заставило обратить особое внимание на службу обеспечения качества: внедрить в производство новые методики тестирования и отбраковки изделий. Однако эти дополнительные затраты, как показала жизнь, полностью окупаются. Винчестеры, выпускаемые фир-

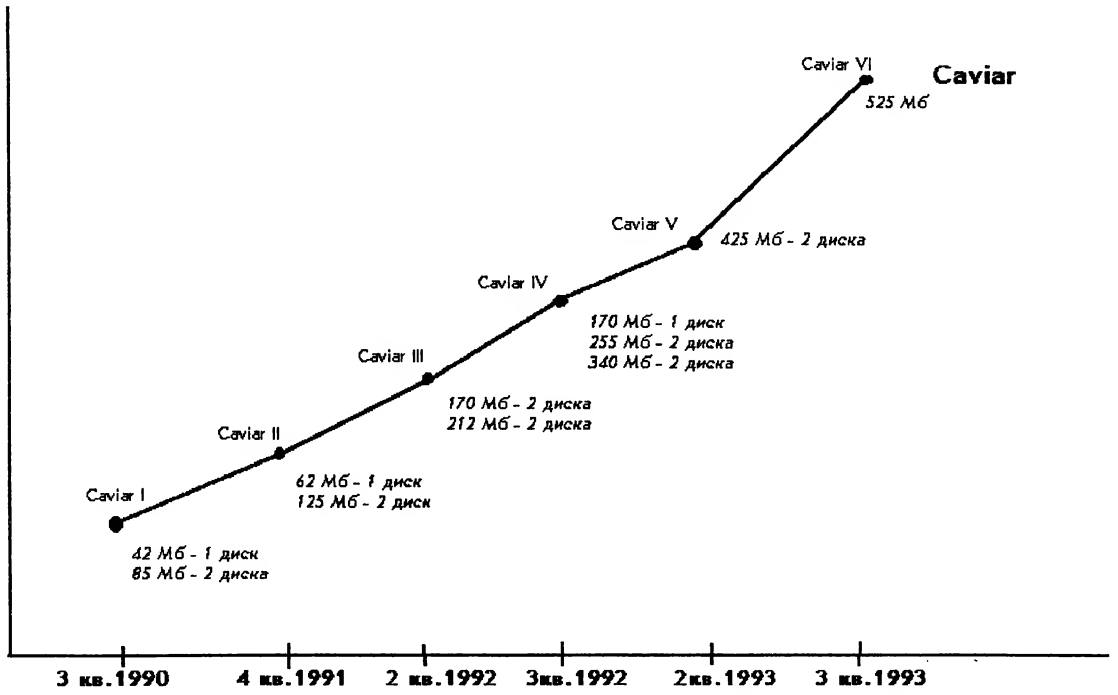


Рис. 1. Динамика развития винчестеров производства фирмы Western Digital

мой Western Digital, имеют время наработки на отказ не менее 100000 часов, что при самых скромных оценках (при работе 10 часов в сутки * 300 дней в году) соответствует приблизительно 33 годам безотказной работы!

Основное стратегическое направление развития винчестеров Western Digital — внедрение последних достижений в области электроники и технологии материалов, что позволяет добиться:

- увеличения емкости накопителей;
- уменьшения массы и габаритов;
- снижения энергопотребления;
- увеличения скорости доступа к данным на диске;
- повышения производительности работы контроллера.

В настоящее время Western Digital разработала две основные серии накопителей: Caviar (3.5") и

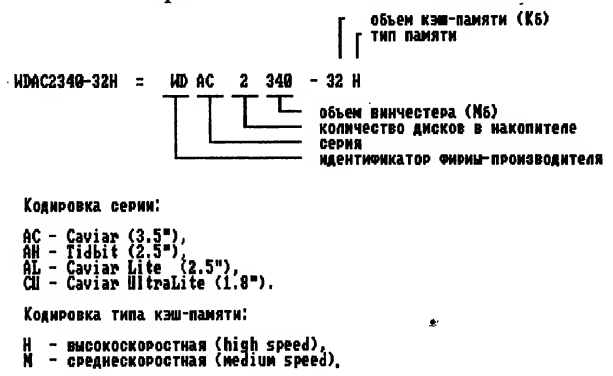


Рис. 2. Кодировка номера модели винчестера производства фирмы Western Digital

TidBit (2.5") и выпустила первую модель винчестера серии Caviar Ultralite (1.8"). На выставке Comdex Fall'92, проходившей в Лас-Вегасе, Western Digital (по заявлению самих представителей фирмы) приложила максимум усилий, чтобы уходящий год запомнился посетителям как год Caviar.

До сих пор слово "Caviar" и у нас, и у американцев ассоциировалось с красивыми стеклянными баночками, заполненными черной икрой. Но, видимо, влияние икры из России на умы и желудки в Соединенных Штатах Америки оказалось столь велико, что новое направление в развитии винчестеров фирмы Western Digital получило название Caviar. Этим названием фирма Western Digital как бы подчеркивает изысканность конструкторских решений, воплощенных в новые разработки, и подчеркивает их ориентированность на взыскательного потребителя. Основная философия Western Digital — идти навстречу нуждам потребителей, изготавливая высококачественную продукцию.

Вообще говоря, серия Caviar заслуживает более пристального рассмотрения. В таблице 1 приведены характеристики каждой из моделей серии, но стоит обратить внимание и на общие для всех изделий параметры (табл. 2).

Приятное впечатление производят диапазон допустимых механических нагрузок и устойчивость к вибрационным нагрузкам. Так, все модели серии Caviar выдерживают перегрузки до 10G в рабочем состоянии (что эквивалентно падению с высоты 10 метров) и до 75G в отключенном состоянии. Остается только позавидовать столь удачным конструкторским решениям. Устойчивость к вибрационным нагрузкам также

Таблица 1

Основные характеристики винчестеров производства фирмы Western Digital

Название	Caviar 80	Caviar 120	Caviar 170	Caviar 200	Caviar 340	Caviar Lite 170	Caviar UltraLite 40
Номер модели	WDAC280	WDAC2120	WDAC2170	WDAC2200	WDAC2340	WDAL2170	WDCU140
Габарит	3.5"	3.5"	3.5"	3.5"	3.5"	2.5"	1.8"
Шина	AT	AT	AT	AT	AT	AT	PCMCIA-ATA
Емкость (Мб) форматированная	85.3	125	170.6	212.6	341.3	170.6	42.7
Время доступа (мс)	< 17	< 15	< 14	< 14	< 13	< 16	< 19
Скорость передачи данных (Мб/с)	4.0	4.5	5.0	5.0	5.75	5.0	4.5
Объем буфера (Кб)	8/32	32	32/64	64	128	32	32
Среднее время наработки на отказ (час)	100000	100000	100000	100000	250000	100000	255000

достаточно высока. Конкретные значения параметров отличаются от модели к модели, и мы приведем здесь усредненные данные:

В рабочем состоянии:

5 — 17 Герц, амплитуда 0.85 мм,

18 — 400 Герц, ускорение 0.75 G.

В отключенном состоянии:

5 — 20 Герц, амплитуда 5.5 мм;

21 — 500 Герц, ускорение 4G.

Следующий момент, на который стоит обратить внимание — это время доступа к данным. Рассмотрим, например, серию 3.5" накопителей. Как следует из таблицы 1, в наихудшем случае (Caviar 80) время доступа не превышает 17 мс (по данным фирмы), а старшие модели показывают еще лучшие результаты. Миниатюрные модели имеют большее время доступа, но это определяется в основном ограничениями механического характера. Однако, надо полагать, в скором времени и модели, ориентированные на использование в notebook компьютерах (Caviar-Lite) и subnotebook компьютерах (Caviar UltraLite), по быстродействию не будут уступать 3.5" собратьям (Вы еще помните о винчестерах со временем доступа 28 мс?).

Таблица 2

Общие характеристики семейства Caviar

Установочный габарит	слот 3.5 дюйма
Длина	5.75 дюйма (146 мм)
Ширина	4.0 дюйма (101.6 мм)
Высота	1.0 дюйм (25.4 мм)
Масса	0.508 кг
Совместимость	AT
Автопарковка	есть
Поддержка Master/Slave	есть

Необходимо отметить, что скорость передачи данных, приведенная в табл. 1, соответствует указанному в табл. 1 объему кэш-памяти. При этом имеется возможность наращивать объем кэш-памяти до 128 Кбайтов (с соответствующим повышением скорости передачи данных), а на новых моделях разработки 1992 года (например, WDAC1170) до 256 Кбайтов.

Интересно, что хорошие показатели времени доступа сочетаются с низким энергопотреблением. Если энергопотребление винчестера FilePro 340 (производитель — фирма Conner из США, ближайший соперник Western Digital на выставке Comdex Fall'92) в режиме ожидания составляет 2.5 ватта, то у винчестера Caviar 340 — всего 1.4 ватта. Это позволяет использовать стандартный блок питания (200 ватт), оснастив компьютер двумя винчестерами, емкостью по 425 Мбайтов каждый.

Очевидно, что для достижения высокого быстродействия и низкого энергопотребления Western Digital пришлось разработать новую механическую часть накопителя. Но на этом инженеры фирмы не остановились. В серии Caviar в моделях разработки 1992 года использован новый контроллер IDE, который также внес свой вклад в улучшение характеристик винчестеров.

Присутствие на российском рынке Western Digital планирует осуществлять только через дистрибьютера, по ряду причин отказываясь от открытия собственного филиала. В настоящее время дистрибьютерское соглашение Western Digital заключено с фирмой "Эксимер".

Познакомиться с продукцией Western Digital Вы сможете на выставке КОМТЕК-93 (Выставочный комплекс на Красной Пресне, стенд 424).

Телефон фирмы "Эксимер": 125-70-01. ■

Мы ставим на лидера!

Компания " Merisel-Компьютерные технологии" - крупнейший дистрибьютер средств вычислительной техники и программных продуктов - расширяет сеть своих дилеров

Для дилеров Компании:

- более 10 000 наименований продукции от 700 производителей;
- отпуск продукции по оптовым ценам независимо от количества приобретаемого оборудования;
- система дополнительных скидок;
- продажа в кредит за рубли и СКВ;
- консультации и рекомендации ведущих специалистов Компании;
- информационно-рекламные материалы ;
- ежегодный форум дилеров Softeach с участием представителей ведущих фирм-производителей.

Вот почему мы стали дилерами MERISEL!

MERISEL
C.A.T.Ltd

109044 Москва, ул. Крутицкий Вал, 3, корп. 2
тел. (095) 276-90-08 (5 линий), факс (095) 274-00-97

CA–Clipper/Compiler Kit

For dBASE IV

Компилируйте и распространяйте dBASE IV-программы с помощью CA-Clipper!

CA–Clipper Compiler Kit for dBASE IV создает из Ваших dBASE IV программ автономные выполняемые .EXE файлы, которые можно свободно распространять как на PC, так и в локальных сетях без дополнительных платежей или Runtime обеспечения.

Компиляция происходит автоматически. Скомпилированные программы полностью поддерживают функциональные возможности dBASE IV и приобретают преимущества, заложенные в CA–Clipper:

— оптимальное управление памятью;
— быстрое выполнение программ;
— доступ ко всем мощным языковым средствам CA–Clipper, включая классы объектов.

Системные требования:

* CA–Clipper (Версия 5.01)
* IBM PC XT, AT, PS/2 или 100% совместимый персональный компьютер
* жесткий диск
* минимум 640K RAM
* драйвер расширенной памяти LIM 3.2 или выше
* DOS 3.1 или выше

CA–Clipper/Compiler Kit For dBASE IV работает в сетях, совместимых с DOS 3.1 или выше

Только три простых шага необходимы для создания автономных программ.

1. Создаете прикладную программу, используя обычный инструментарий dBASE IV, включая генератор прикладных программ.
2. Оптимизируете и тестируете Вашу программу.
3. Набираете команду COMPILE имя файла

То, что Ashton Tate и Borland обещали шесть лет, Computer Associates предлагает Вам сегодня!

127108, Москва, ул. 2–я Ямская, д. 15

Телефоны: 289–44–77, 289–44–83

COMPUTER ASSOCIATES®
Software superior by design